PYTHON数据科学系列课程（三）
NUMPY案例及PANDAS入门

东南大学 学习科学研究中心 儿童发展与教育研究所

夏小俊

http://www.seuct.com

NUMPY案例讲解

- 股票数据分析
- 简单图像操作

本章内容提要

- 3.1 Pandas简介
- 3.2 Series的定义方法
- 3.3 Series的操作方法
- 3.4 Series的修改

3.1 PANDAS简介

Pandas是python的一个数据分析包，基于Numpy而设计。最初由AQR Capital Management于2008年4月开发，并于2009年底开源出来，目前由专注于Python数据包开发的PyData开发team继续开发和维护，属于PyData项目的一部分。

Pandas最初被作为金融数据分析工具而开发出来，名称来自于面板数据（panel data）和python数据分析（data analysis）。panel data是经济学中关于多维数据集的一个术语，在Pandas中也提供了panel的数据类型。

Pandas提供了序列Series和数据库DataFrame两种核心的数据结构，操作简便功能强大，是学习数据科学必不可少的内容。

通过pip install pandas即可安装。

3.2 SERIES(序列)的定义方法

In[1]:

```
import pandas as pd
mySeries1=pd.Series(data =
[11,12,13,14,15,16,17],index=["a","b","c","d","e","f","g"])
mySeries1
```

Out[1]:

```
a    11
b    12
c    13
d    14
e    15
f    16
g    17
dtype: int64
```

序列:用于存储一行或一列的数据,以及与之相对应的索引的集合.

3.2 SERIES的定义方法

In[2]:

```
import pandas as pd
mySeries1=pd.Series([11,12,13,14,15,16,17],
index=[a,b,c,d,e,f,g])
mySeries1
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-3-88cdcb222886> in <module>()
      1 import pandas as pd
----> 2 mySeries1=pd.Series([11,12,13,14,15,16,17],
index=[a,b,c,d,e,f,g])
      3 mySeries1
NameError: name 'a' is not defined
```

3.2 SERIES的定义方法

In[3]:

```
mySeries2=pd.Series(10,
index=["a","b","c","d","e","f","g"])
mySeries2
```

Out[3]:

```
a    10
b    10
c    10
d    10
e    10
f    10
g    10
dtype: int64
```

3.2 SERIES的定义方法

In[4]:

```
mySeries3=pd.Series([1,2,3,4,5], index=["a","b","c"])
mySeries3
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-5-e8a37d3e2b30> in <module>()
----> 1 mySeries3=pd.Series([1,2,3,4,5], index=["a","b","c"])
      2 mySeries3
......
---> 90                                 len(self.mgr_locs)))
     91
     92     @property

ValueError: Wrong number of items passed 5, placement implies 3
```

3.3 SERIES的操作方法

In[5]:
```
import pandas as pd
mySeries4=pd.Series([21,22,23,24,25,26,27],
index=["a","b","c","d","e","f","g"])
mySeries4.index
```
Out[5]: Index(['a', 'b', 'c', 'd', 'e', 'f', 'g'], dtype='object')

In[6]:
```
mySeries4.values  #numpy对象
```
Out[6]: array([21, 22, 23, 24, 25, 26, 27], dtype=int64)

In[7]:
```
mySeries4['b']
```
Out[7]: 22

3.3 SERIES的操作方法

In[8]:
```
mySeries4[["a","b","c"]]
```

Out[8]:
```
a    21
b    22
c    23
dtype: int64
```

37.3 SERIES的操作方法

In[9]:
```
mySeries4[“a”:“d”] #注意和数字索引切片的区别
```

Out[9]:
```
a    21
b    22
c    23
d    24
dtype: int64
```

In[10]:
```
mySeries4[1:4:2]
```

Out[10]:
```
b    22
d    24
dtype: int64
```

3.3 SERIES的操作方法

In[11]:
```
"c" in mySeries4
```

Out[11]: **True**

In[12]:
```
"h" in mySeries4
```

Out[12]: **False**

In[13]:
```
mySeries4.isin([22 , 23]) #isin函数
```

3.3 SERIES的操作方法

In[13]:

```
mySeries4=pd.Series([21,22,23,24,25,26,27],
index=["a","b","c","d","e","f","g"])
mySeries5=mySeries4.reindex(index=["b","c","a","d","e","g
","f"])
mySeries5
```

Out[13]:

```
b    22
c    23
a    21
d    24
e    25
g    27
f    26
dtype: int64
```

3.3 SERIES的操作方法

In[14]:
```
mySeries5=mySeries4.reindex(index=["new1","c","a","new2","e","g","new3"])
mySeries5
```

Out[14]:
```
new1    NaN
c       23.0
a       21.0
new2    NaN
e       25.0
g       27.0
new3    NaN
dtype: float64
```

NaN : Not a Number

3.4 SERIES的修改

In[15]:

```
mySeries5=pd.Series([28])

mySeries6 = mySeries4.append(mySeries5 ,
ignore_index=False) #是否忽略索引
```

3.4 SERIES的修改

In[16]:

```
mySeries4['a'] = 28
mySeries4[['b','d']] = [30,31]
mySeries4
```

3.4 SERIES的修改

In[17]:

```
mySeries5 = mySeries4.drop(['a','b'])
mySeries6 = mySeries4.drop('a')
```

3.5 创建DATAFRAME

In[1]:

```
import numpy as np
import pandas as pd
df1=pd.DataFrame(np.arange(10).reshape(2,5))
df1
```

Out[1]:

	0	1	2	3	4
0	0	1	2	3	4
1	5	6	7	8	9

In[2]:

```
df2 = pd.read_csv('shopping.csv')
df2.shape
```

Out[2]: (10, 6)

3.5 创建DATAFRAME

In[3]:

```
df3=pd.DataFrame(
    data = { #字典
        'name' : ['tom' , 'jack' , 'mary'],
        'age' : [20 , 18 , 16]
    },
    index = [‘first’ , ‘second’ , ‘third’], #可选,指定索引
    columns = [‘age’ , ‘name‘] #可选,指定列的名字和顺序
)
```

3.5 创建DATAFRAME

In[4]:

```
df2 = df2[["id","date","money"]]
df2.head() #前五条数据
```

Out[4]:

3.6 查看行或列

In[5]:
```
df2.index
```
Out[5]: RangeIndex(start=0, stop=10, step=1)

In[6]:
```
df2.index.size
```
Out[6]: 10

In[7]:
```
df2.columns
```
Out[7]: Index(['id', 'date', 'money'], dtype='object')

3.6 查看行或列

In[8]:
```
df2.columns.size
```
Out[8]: 3

In[9]:
```
df2.shape
```
Out[9]: (10, 3)

In[10]:
```
print("行数为:", df2.shape[0])
print("列数为:", df2.shape[1])
```
Out[10]: 行数为: 10
列数为: 3

3.6 查看行或列

In[11]:
```
df2["id"].head()
```

Out[11]:
```
0    101
1    102
2    103
3    104
4    105
Name: id, dtype: int64
```

3.6 查看行或列

In[12]:
```
df2.id.head() #向量化
```

Out[12]:
```
0    101
1    102
2    103
3    104
4    105
Name: id, dtype: int64
```

In[13]:
```
df2["id"][2]
```

Out[13]:
```
103
```

3.6 查看行或列

In[14]:
```
df2.id[2]
```

Out[14]: 103

In[15]:
```
df2["id"][[2,4]]
```

Out[15]:
```
2    103
3    4    105
4    Name: id, dtype: int64
```

In[16]:
```
df2.loc[1,“id”] #loc(自定义)和iloc(默认)
```

Out[16]: 102

3.6 查看行或列

In[17]:
```
df2.iloc[1,0]
```

Out[17]: 102

In[18]:
```
df2.loc[1:5,["id"]]
df2.iloc[1:5,0] #注意两种方式当中切片的区别!
```

3.6 查看行或列

In[19]:

```
df2[["date","id"]].head()
```

Out[19]:

In[20]:

```
df3 = df2.iloc[0:3]

df3.reindex(index=[2,0,1] , columns = ['money' , 'id' , 'date'])
```

Out[20]:

3.7 修改或过滤行/列

In[21]:

```
df2.drop(0) #删除行
df2.drop(['id'] , axis = 1) #删除列
```

In[22]:

```
df2.drop([3,4], axis=0, inplace=True) #就地修改
df2.head()
```

3.7 修改或过滤行/列

In[23]:

```
import pandas as pd
df2 = pd.read_csv('shopping.csv')
df2 = df2[["id","date","money"]]
df2[df2.money > 10]
```

In[24]:

```
df2 = pd.read_csv('shopping.csv')
df1 = df2[["origin","product"]].head()
df3 = df2[["id","date","money"]].head()
df4 = df1.append(df3)
df4.loc[0]
```

3.7 修改或过滤行/列

In[25]:

```
df2 = pd.read_csv('shopping.csv')

df2 = df2[["id","date","money"]].head()
df2[‘dd’] = [1,2,3,4,5] #新增列
df2
```

In[26]:

```
df2.loc[0:1,'id'] = [107,108]

df2.at[[0,1] , 'id'] = [110,111]
```

3.8 算术运算

In[1]:

```
df4=pd.DataFrame(np.arange(6).reshape(2,3))
df4
```

Out[1]:

	0	1	2
0	0	1	2
1	3	4	5

In[2]:

```
df5=pd.DataFrame(np.arange(10).reshape(2,5))
df5
```

Out[2]:

	0	1	2	3	4
0	0	1	2	3	4
1	5	6	7	8	9

3.8 算术运算

In[3]:

```
df4+df5
```

Out[3]:

	0	1	2	3	4
0	0	2	4	NaN	NaN
1	8	10	12	NaN	NaN

In[4]:

```
df6=df4.add(df5,fill_value=10)
df6
```

Out[4]:

	0	1	2	3	4
0	0	2	4	13.0	14.0
1	8	10	12	18.0	19.0

3.8 算术运算

In[5]:
```
sl=pd.Series(np.arange(3))
sl
```

Out[5]:
```
0    0
1    1
2    2
dtype: int32
```

In[6]:
```
df6-sl
```

Out[6]:

	0	1	2	3	4
0	0.0	1.0	2.0	NaN	NaN
1	8.0	9.0	10.0	NaN	NaN

3.8 算术运算

In[6]:

```
df5=pd.DataFrame(np.arange(10).reshape(2,5))
s1=pd.Series(np.arange(3))
df5-s1
```

Out[6]:

	0	1	2	3	4
0	0.0	0.0	0.0	NaN	NaN
1	5.0	5.0	5.0	NaN	NaN

3.8 算术运算

In[7]:

```
df5=pd.DataFrame(np.arange(10).reshape(2,5))
s1=pd.Series(np.arange(3))
df5.sub(s1,axis=1) #广播
```

Out[7]:

	0	1	2	3	4
0	0.0	0.0	0.0	NaN	NaN
1	5.0	5.0	5.0	NaN	NaN

3.8 算术运算

In[8]:

```
df5=pd.DataFrame(np.arange(10).reshape(2,5))
s1=pd.Series(np.arange(3))
df5.sub(s1,axis=0)
```

Out[8]:

	0	1	2	3	4
0	0.0	1.0	2.0	3.0	4.0
1	4.0	5.0	6.0	7.0	8.0
2	NaN	NaN	NaN	NaN	NaN

3.8 算术运算

In[9]:

```
df7=pd.DataFrame(np.arange(20).reshape(4,5))
df7
```

Out[9]:

	0	1	2	3	4
0	0	1	2	3	4
1	5	6	7	8	9
2	10	11	12	13	14
3	15	16	17	18	19

3.8 算术运算

In[10]:

```
df7+2
```

Out[10]:

	0	1	2	3	4
0	2	3	4	5	6
1	7	8	9	10	11
2	12	13	14	15	16
3	17	18	19	20	21

3.8 算术运算

In[11]:

```
print(df7)
print("df7.cumsum=",df7.cumsum()) #累加求和
```

```
   0  1  2  3  4
0  0  1  2  3  4
1  5  6  7  8  9
2 10 11 12 13 14
3 15 16 17 18 19
df7.cumsum=    0  1  2  3  4
0  0  1  2  3  4
1  5  7  9 11 13
2 15 18 21 24 27
3 30 34 38 42 46
```

3.8 算术运算

In[12]:

```
df7.rolling(2).sum() #时间窗函数
```

Out[12]:

	0	1	2	3	4
0	NaN	NaN	NaN	NaN	NaN
1	5.0	7.0	9.0	11.0	13.0
2	15.0	17.0	19.0	21.0	23.0
3	25.0	27.0	29.0	31.0	33.0

3.8 算术运算

In[13]:

```
df7.rolling(2,axis=1).sum()
```

Out[13]:

	0	1	2	3	4
0	NaN	1.0	3.0	5.0	7.0
1	NaN	11.0	13.0	15.0	17.0
2	NaN	21.0	23.0	25.0	27.0
3	NaN	31.0	33.0	35.0	37.0

3.8 算术运算

In[14]:

```
df7.cov() #协方差矩阵
```

Out[14]:

	0	1	2	3	4
0	41.666667	41.666667	41.666667	41.666667	41.666667
1	41.666667	41.666667	41.666667	41.666667	41.666667
2	41.666667	41.666667	41.666667	41.666667	41.666667
3	41.666667	41.666667	41.666667	41.666667	41.666667
4	41.666667	41.666667	41.666667	41.666667	41.666667

3.8 算术运算

In[15]:

```
df7.corr() #相关系数矩阵
```

Out[15]:

	0	1	2	3	4
0	1.0	1.0	1.0	1.0	1.0
1	1.0	1.0	1.0	1.0	1.0
2	1.0	1.0	1.0	1.0	1.0
3	1.0	1.0	1.0	1.0	1.0
4	1.0	1.0	1.0	1.0	1.0

谢谢大家！