1.Pandas概述

1. Pandas是Python的一个数据分析包，该工具为解决数据分析任务而创建。
2. Pandas纳入大量库和标准数据模型，提供高效的操作数据集所需的工具。
3. Pandas提供大量能使我们快速便捷地处理数据的函数和方法。
4. Pandas是字典形式，基于NumPy创建，让NumPy为中心的应用变得更加简单。

2.Pandas安装

```
pip3 install pandas
```

3.Pandas引入

```
import pandas as pd#为了方便实用pandas 采用pd简写
```

4.Pandas数据结构

4.1Series

```
import numpy as np
import pandas as pd
s=pd.Series([1,2,3,np.nan,5,6])
print(s)#索引在左边 值在右边
'''
0    1.0
1    2.0
2    3.0
3    NaN
4    5.0
5    6.0
dtype: float64
 '''
```

4.2DataFrame

DataFrame是表格型数据结构，包含一组有序的列，每列可以是不同的值类型。DataFrame有行索引和列索引，可以看成由Series组成的字典。

```
dates=pd.date_range('20180310',periods=6)
df = pd.DataFrame(np.random.randn(6,4), index=dates, columns=
['A','B','C','D'])#生成6行4列位置
print(df)#输出6行4列的表格
'''
```

```
                   A         B         C         D
2018-03-10 -0.092889 -0.503172  0.692763 -1.261313
2018-03-11 -0.895628 -2.300249 -1.098069  0.468986
2018-03-12  0.084732 -1.275078  1.638007 -0.291145
2018-03-13 -0.561528  0.431088  0.430414  1.065939
2018-03-14  1.485434 -0.341404  0.267613 -1.493366
2018-03-15 -1.671474  0.110933  1.688264 -0.910599
  '''
print(df['B'])
'''
2018-03-10   -0.927291
2018-03-11   -0.406842
2018-03-12   -0.088316
2018-03-13   -1.631055
2018-03-14   -0.929926
2018-03-15   -0.010904
Freq: D, Name: B, dtype: float64
 '''

#创建特定数据的DataFrame
df_1=pd.DataFrame({'A' : 1.,
                    'B' : pd.Timestamp('20180310'),
                    'C' :
pd.Series(1,index=list(range(4)),dtype='float32'),
                    'D' : np.array([3] * 4,dtype='int32'),
                    'E' : pd.Categorical(["test","train","test","train"]),
                    'F' : 'foo'
                    })
print(df_1)
'''
     A          B    C  D      E    F
0  1.0 2018-03-10  1.0  3   test  foo
1  1.0 2018-03-10  1.0  3  train  foo
2  1.0 2018-03-10  1.0  3   test  foo
3  1.0 2018-03-10  1.0  3  train  foo
'''
print(df_1.dtypes)
'''
A           float64
B    datetime64[ns]
C           float32
D             int32
E          category
F            object
dtype: object
'''
print(df_1.index)#行的序号
#Int64Index([0, 1, 2, 3], dtype='int64')
print(df_1.columns)#列的序号名字
```

```
#Index(['A', 'B', 'C', 'D', 'E', 'F'], dtype='object')
print(df_1.values)#把每个值进行打印出来
'''
[[1.0 Timestamp('2018-03-10 00:00:00') 1.0 3 'test' 'foo']
 [1.0 Timestamp('2018-03-10 00:00:00') 1.0 3 'train' 'foo']
 [1.0 Timestamp('2018-03-10 00:00:00') 1.0 3 'test' 'foo']
 [1.0 Timestamp('2018-03-10 00:00:00') 1.0 3 'train' 'foo']]
 '''
print(df_1.describe())#数字总结
'''
         A    C    D
count  4.0  4.0  4.0
mean   1.0  1.0  3.0
std    0.0  0.0  0.0
min    1.0  1.0  3.0
25%    1.0  1.0  3.0
50%    1.0  1.0  3.0
75%    1.0  1.0  3.0
max    1.0  1.0  3.0
'''
print(df_1.T)#翻转数据
'''
                     0                    1                    2  \
A                    1                    1                    1
B  2018-03-10 00:00:00  2018-03-10 00:00:00  2018-03-10 00:00:00
C                    1                    1                    1
D                    3                    3                    3
E                 test                train                 test
F                  foo                  foo                  foo

                     3
A                    1
B  2018-03-10 00:00:00
C                    1
D                    3
E                train
F                  foo
'''
print(df_1.sort_index(axis=1, ascending=False))#axis等于1按列进行排序 如ABCDEFG 然后ascending倒叙进行显示
'''
     F      E  D    C          B    A
0  foo   test  3  1.0 2018-03-10  1.0
1  foo  train  3  1.0 2018-03-10  1.0
2  foo   test  3  1.0 2018-03-10  1.0
3  foo  train  3  1.0 2018-03-10  1.0
'''
print(df_1.sort_values(by='E'))#按值进行排序
'''
```

```
     A          B    C  D      E    F
0  1.0 2018-03-10  1.0  3   test  foo
2  1.0 2018-03-10  1.0  3   test  foo
1  1.0 2018-03-10  1.0  3  train  foo
3  1.0 2018-03-10  1.0  3  train  foo
'''
```

###5.Pandas选择数据

```
dates=pd.date_range('20180310',periods=6)
df = pd.DataFrame(np.random.randn(6,4), index=dates, columns=
['A','B','C','D'])#生成6行4列位置
print(df)
'''
                   A         B         C         D
2018-03-10 -0.520509 -0.136602 -0.516984  1.357505
2018-03-11  0.332656 -0.094633  0.382384 -0.914339
2018-03-12  0.499960  1.576897  2.128730  2.197465
2018-03-13  0.540385  0.427337 -0.591381  0.126503
2018-03-14  0.191962  1.237843  1.903370  2.155366
2018-03-15 -0.188331 -0.578581 -0.845854 -0.056373
 '''
print(df['A'])#或者df.A 选择某列
'''
2018-03-10   -0.520509
2018-03-11    0.332656
2018-03-12    0.499960
2018-03-13    0.540385
2018-03-14    0.191962
2018-03-15   -0.188331
'''
```

切片选择

```
print(df[0:3], df['20180310':'20180314'])#两次进行选择 第一次切片选择 第二次按照
筛选条件进行选择
'''
                   A         B         C         D
2018-03-10 -0.520509 -0.136602 -0.516984  1.357505
2018-03-11  0.332656 -0.094633  0.382384 -0.914339
2018-03-12  0.499960  1.576897  2.128730  2.197465
                  A         B         C         D
2018-03-10 -0.520509 -0.136602 -0.516984  1.357505
2018-03-11  0.332656 -0.094633  0.382384 -0.914339
2018-03-12  0.499960  1.576897  2.128730  2.197465
2018-03-13  0.540385  0.427337 -0.591381  0.126503
2018-03-14  0.191962  1.237843  1.903370  2.155366
 '''
```

根据标签loc-行标签进行选择数据

```
print(df.loc['20180312', ['A','B']])#按照行标签进行选择 精确选择
 '''
A    0.499960
B    1.576897
Name: 2018-03-12 00:00:00, dtype: float64
'''
```

根据序列iloc-行号进行选择数据

```
print(df.iloc[3, 1])#输出第三行第一列的数据
#0.427336827399

print(df.iloc[3:5,0:2])#进行切片选择
 '''
                   A         B
2018-03-13  0.540385  0.427337
2018-03-14  0.191962  1.237843
 '''


print(df.iloc[[1,2,4],[0,2]])#进行不连续筛选
'''
                   A         C
2018-03-11  0.332656  0.382384
2018-03-12  0.499960  2.128730
2018-03-14  0.191962  1.903370
 '''
```

根据混合的两种ix

```
print(df.ix[:3, ['A', 'C']])
'''
                   A         C
2018-03-10 -0.919275 -1.356037
2018-03-11  0.010171 -0.380010
2018-03-12  0.285251 -1.174265
 '''
```

根据判断筛选

```
print(df[df.A > 0])#筛选出df.A大于0的元素 布尔条件筛选
'''
                   A         B         C         D
2018-03-11  0.332656 -0.094633  0.382384 -0.914339
2018-03-12  0.499960  1.576897  2.128730  2.197465
2018-03-13  0.540385  0.427337 -0.591381  0.126503
2018-03-14  0.191962  1.237843  1.903370  2.155366
 '''
```

6.Pandas设置数据

根据loc和iloc设置

```
dates = pd.date_range('20180310', periods=6)
df = pd.DataFrame(np.arange(24).reshape((6,4)), index=dates, columns=['A',
'B', 'C', 'D'])
print(df)
'''
             A   B     C   D
2018-03-10   0   1     2   3
2018-03-11   4   5     6   7
2018-03-12   8   9  1111  11
2018-03-13  12  13    14  15
2018-03-14  16  17    18  19
2018-03-15  20  21    22  23
'''

df.iloc[2,2] = 999#单点设置
df.loc['2018-03-13', 'D'] = 999
print(df)
'''
            A   B    C    D
2018-03-10  0   1    2    3
2018-03-11  0   5    6    7
2018-03-12  0   9  999   11
2018-03-13  0  13   14  999
2018-03-14  0  17   18   19
2018-03-15  0  21   22   23
'''
```

根据条件设置

```
df[df.A>0]=999#将df.A大于0的值改变
print(df)
'''
              A   B    C    D
2018-03-10    0   1    2    3
2018-03-11  999   5    6    7
2018-03-12  999   9  999   11
2018-03-13  999  13   14  999
2018-03-14  999  17   18   19
2018-03-15  999  21   22   23
 '''
```

根据行或列设置

```
df['F']=np.nan
print(df)
'''
              A   B    C   D
2018-03-10    0   1    2 NaN
2018-03-11  999   5    6 NaN
2018-03-12  999   9  999 NaN
2018-03-13  999  13   14 NaN
2018-03-14  999  17   18 NaN
2018-03-15  999  21   22 NaN
 '''
```

添加数据

```
df['E']  = pd.Series([1,2,3,4,5,6], index=pd.date_range('20180313', periods=6))#增加一列
print(df)
'''
              A   B    C   D    E
2018-03-10    0   1    2 NaN  NaN
2018-03-11  999   5    6 NaN  NaN
2018-03-12  999   9  999 NaN  NaN
2018-03-13  999  13   14 NaN  1.0
2018-03-14  999  17   18 NaN  2.0
2018-03-15  999  21   22 NaN  3.0
'''
```

7.Pandas处理丢失数据

处理数据中NaN数据

```
dates = pd.date_range('20180310', periods=6)
```

```
df = pd.DataFrame(np.arange(24).reshape((6,4)), index=dates, columns=['A',
'B', 'C', 'D'])
df.iloc[0,1]=np.nan
df.iloc[1,2]=np.nan
print(df)
'''
             A     B     C   D
2018-03-10   0   NaN   2.0   3
2018-03-11   4   5.0   NaN   7
2018-03-12   8   9.0  10.0  11
2018-03-13  12  13.0  14.0  15
2018-03-14  16  17.0  18.0  19
2018-03-15  20  21.0  22.0  23
'''
```

使用dropna（）函数去掉NaN的行或列

```
print(df.dropna(axis=0,how='any'#))#0对行进行操作 1对列进行操作 any:只要存在NaN
即可drop掉 all:必须全部是NaN才可drop
'''
             A     B     C   D
2018-03-12   8   9.0  10.0  11
2018-03-13  12  13.0  14.0  15
2018-03-14  16  17.0  18.0  19
2018-03-15  20  21.0  22.0  23
 '''
```

使用fillna（）函数替换NaN值

```
print(df.fillna(value=0))#将NaN值替换为0
'''
             A     B     C   D
2018-03-10   0   0.0   2.0   3
2018-03-11   4   5.0   0.0   7
2018-03-12   8   9.0  10.0  11
2018-03-13  12  13.0  14.0  15
2018-03-14  16  17.0  18.0  19
2018-03-15  20  21.0  22.0  23
 '''
```

使用isnull()函数判断数据是否丢失

```
print(pd.isnull(df))#矩阵用布尔来进行表示 是nan为ture 不是nan为false
'''
                A      B      C      D
2018-03-10  False   True  False  False
2018-03-11  False  False   True  False
2018-03-12  False  False  False  False
2018-03-13  False  False  False  False
2018-03-14  False  False  False  False
2018-03-15  False  False  False  False
 '''
print(np.any(df.isnull()))#判断数据中是否会存在NaN值
#True
```

8.Pandas导入导出

pandas可以读取与存取像csv、excel、json、html、pickle等格式的资料，详细说明请看官方资料

```
data=pd.read_csv('test1.csv')#读取csv文件
data.to_pickle('test2.pickle')#将资料存取成pickle文件
#其他文件导入导出方式相同
```

9.Pandas合并数据

axis合并方向

```
df1 = pd.DataFrame(np.ones((3,4))*0, columns=['a','b','c','d'])
df2 = pd.DataFrame(np.ones((3,4))*1, columns=['a','b','c','d'])
df3 = pd.DataFrame(np.ones((3,4))*2, columns=['a','b','c','d'])
res = pd.concat([df1, df2, df3], axis=0, ignore_index=True)#0表示竖项合并 1表示横项合并 ingnore_index重置序列index index变为0 1 2 3 4 5 6 7 8
print(res)
'''
     a    b    c    d
0  0.0  0.0  0.0  0.0
1  0.0  0.0  0.0  0.0
2  0.0  0.0  0.0  0.0
3  1.0  1.0  1.0  1.0
4  1.0  1.0  1.0  1.0
5  1.0  1.0  1.0  1.0
6  2.0  2.0  2.0  2.0
7  2.0  2.0  2.0  2.0
8  2.0  2.0  2.0  2.0
 '''
```

join合并方式

```
df1 = pd.DataFrame(np.ones((3,4))*0, columns=['a','b','c','d'], index=
[1,2,3])
df2 = pd.DataFrame(np.ones((3,4))*1, columns=['b','c','d', 'e'], index=
[2,3,4])
print(df1)
'''
     a    b    c    d
1  0.0  0.0  0.0  0.0
2  0.0  0.0  0.0  0.0
3  0.0  0.0  0.0  0.0
 '''
print(df2)
'''
     b    c    d    e
2  1.0  1.0  1.0  1.0
3  1.0  1.0  1.0  1.0
4  1.0  1.0  1.0  1.0
 '''
res=pd.concat([df1,df2],axis=1,join='outer')#行往外进行合并
print(res)
'''
     a    b    c    d    b    c    d    e
1  0.0  0.0  0.0  0.0  NaN  NaN  NaN  NaN
2  0.0  0.0  0.0  0.0  1.0  1.0  1.0  1.0
3  0.0  0.0  0.0  0.0  1.0  1.0  1.0  1.0
4  NaN  NaN  NaN  NaN  1.0  1.0  1.0  1.0
 '''

res=pd.concat([df1,df2],axis=1,join='outer')#行相同的进行合并
print(res)
'''
     a    b    c    d    b    c    d    e
2  0.0  0.0  0.0  0.0  1.0  1.0  1.0  1.0
3  0.0  0.0  0.0  0.0  1.0  1.0  1.0  1.0
'''

res=pd.concat([df1,df2],axis=1,join_axes=[df1.index])#以df1的序列进行合并 df2
中没有的序列NaN值填充
print(res)
'''
     a    b    c    d    b    c    d    e
1  0.0  0.0  0.0  0.0  NaN  NaN  NaN  NaN
2  0.0  0.0  0.0  0.0  1.0  1.0  1.0  1.0
3  0.0  0.0  0.0  0.0  1.0  1.0  1.0  1.0
'''
```

append添加数据

```
df1 = pd.DataFrame(np.ones((3,4))*0, columns=['a','b','c','d'])
df2 = pd.DataFrame(np.ones((3,4))*1, columns=['a','b','c','d'])
df3 = pd.DataFrame(np.ones((3,4))*1, columns=['a','b','c','d'])
s1 = pd.Series([1,2,3,4], index=['a','b','c','d'])

res=df1.append(df2,ignore_index=True)#将df2合并到df1的下面 并重置index
print(res)
'''
     a    b    c    d
0  0.0  0.0  0.0  0.0
1  0.0  0.0  0.0  0.0
2  0.0  0.0  0.0  0.0
3  1.0  1.0  1.0  1.0
4  1.0  1.0  1.0  1.0
5  1.0  1.0  1.0  1.0
'''

res=df1.append(s1,ignore_index=True)#将s1合并到df1下面 并重置index
print(res)
'''
     a    b    c    d
0  0.0  0.0  0.0  0.0
1  0.0  0.0  0.0  0.0
2  0.0  0.0  0.0  0.0
3  1.0  2.0  3.0  4.0
'''
```

10.Pandas合并merge

依据一组key合并

```
left = pd.DataFrame({'key': ['K0', 'K1', 'K2', 'K3'],
                     'A': ['A0', 'A1', 'A2', 'A3'],
                     'B': ['B0', 'B1', 'B2', 'B3']})
print(left)
'''
    A   B key
0  A0  B0  K0
1  A1  B1  K1
2  A2  B2  K2
3  A3  B3  K3
'''
right = pd.DataFrame({'key': ['K0', 'K1', 'K2', 'K3'],
                      'C': ['C0', 'C1', 'C2',  'C3'],
                      'D': ['D0', 'D1', 'D2', 'D3']})
print(right)
'''
    C   D key
```

```
0  C0  D0  K0
1  C1  D1  K1
2  C2  D2  K2
3  C3  D3  K3
'''
res=pd.merge(left,right,on='key')
print(res)
'''
    A   B key   C   D
0  A0  B0  K0  C0  D0
1  A1  B1  K1  C1  D1
2  A2  B2  K2  C2  D2
3  A3  B3  K3  C3  D3
'''
```

依据两组key合并

```
left = pd.DataFrame({'key1': ['K0', 'K0', 'K1', 'K2'],
                             'key2': ['K0', 'K1', 'K0', 'K1'],
                             'A': ['A0', 'A1', 'A2', 'A3'],
                             'B': ['B0', 'B1', 'B2', 'B3']})
print(left)
'''
    A   B key1 key2
0  A0  B0   K0   K0
1  A1  B1   K0   K1
2  A2  B2   K1   K0
3  A3  B3   K2   K1
 '''
right = pd.DataFrame({'key1': ['K0', 'K1', 'K1', 'K2'],
                              'key2': ['K0', 'K0', 'K0', 'K0'],
                              'C': ['C0', 'C1', 'C2', 'C3'],
                              'D': ['D0', 'D1', 'D2', 'D3']})
print(right)
'''
    C   D key1 key2
0  C0  D0   K0   K0
1  C1  D1   K1   K0
2  C2  D2   K1   K0
3  C3  D3   K2   K0
 '''

res=pd.merge(left,right,on=['key1','key2'],how='inner')#内联合并
print(res)
'''
    A   B key1 key2   C   D
0  A0  B0   K0   K0  C0  D0
1  A2  B2   K1   K0  C1  D1
2  A2  B2   K1   K0  C2  D2
```

```
'''

res=pd.merge(left,right,on=['key1','key2'],how='outer')#外联合并
print(res)
'''
     A    B key1 key2    C    D
0   A0   B0   K0   K0   C0   D0
1   A1   B1   K0   K1  NaN  NaN
2   A2   B2   K1   K0   C1   D1
3   A2   B2   K1   K0   C2   D2
4   A3   B3   K2   K1  NaN  NaN
5  NaN  NaN   K2   K0   C3   D3
'''

res=pd.merge(left,right,on=['key1','key2'],how='left')#左联合并
'''
    A   B key1 key2    C    D
0  A0  B0   K0   K0   C0   D0
1  A1  B1   K0   K1  NaN  NaN
2  A2  B2   K1   K0   C1   D1
3  A2  B2   K1   K0   C2   D2
4  A3  B3   K2   K1  NaN  NaN
'''

res=pd.merge(left,right,on=['key1','key2'],how='right')#右联合并
print(res)
'''
     A    B key1 key2   C   D
0   A0   B0   K0   K0  C0  D0
1   A2   B2   K1   K0  C1  D1
2   A2   B2   K1   K0  C2  D2
3  NaN  NaN   K2   K0  C3  D3
'''
```

Indicator合并

```
df1 = pd.DataFrame({'col1':[0,1], 'col_left':['a','b']})
print(df1)
'''
   col1 col_left
0     0        a
1     1        b
 '''
df2 = pd.DataFrame({'col1':[1,2,2],'col_right':[2,2,2]})
print(df2)
'''
   col1  col_right
0     1          2
1     2          2
```

```
2     2          2
 '''

res=pd.merge(df1,df2,on='col1',how='outer',indicator=True)#依据col1进行合并并启用indicator=True输出每项合并方式
print(res)
'''
   col1 col_left  col_right      _merge
0     0        a        NaN   left_only
1     1        b        2.0        both
2     2      NaN        2.0  right_only
3     2      NaN        2.0  right_only
'''

res = pd.merge(df1, df2, on='col1', how='outer',
indicator='indicator_column')#自定义indicator column名称
print(res)
'''
   col1 col_left  col_right indicator_column
0     0        a        NaN        left_only
1     1        b        2.0             both
2     2      NaN        2.0       right_only
3     2      NaN        2.0       right_only
'''
```

依据index合并

```
left = pd.DataFrame({'A': ['A0', 'A1', 'A2'],
                                  'B': ['B0', 'B1', 'B2']},
                                  index=['K0', 'K1', 'K2'])
print(left)
'''
     A   B
K0  A0  B0
K1  A1  B1
K2  A2  B2
 '''
right = pd.DataFrame({'C': ['C0', 'C2', 'C3'],
                                     'D': ['D0', 'D2', 'D3']},
                                      index=['K0', 'K2', 'K3'])
print(right)
'''
     C   D
K0  C0  D0
K2  C2  D2
K3  C3  D3
'''
```

```
res=pd.merge(left,right,left_index=True,right_index=True,how='outer')#根据
index索引进行合并 并选择外联合并
print(res)
'''
      A    B    C    D
K0   A0   B0   C0   D0
K1   A1   B1  NaN  NaN
K2   A2   B2   C2   D2
K3  NaN  NaN   C3   D3
'''


res=pd.merge(left,right,left_index=True,right_index=True,how='inner')
print(res)
'''
     A   B   C   D
K0  A0  B0  C0  D0
K2  A2  B2  C2  D2
'''
```

更多内容请关注公众号'谓之小一'，若有疑问可在公众号后台提问，随时回答，内容转载请注明出处。「谓之小一」希望提供给读者别处看不到的内容，关于互联网、机器学习、数据挖掘、编程、书籍、生活...

「谓之小一」希望提供给读者别处看不到的内容，关于互联网、数据挖掘、机器学习、书籍、生活……

- 知乎：@谓之小一
- 公众号：@谓之小一
- GitHub：@weizhixiaoyi
- 技术博客：https://weizhixiaoyi.com

长按关注微信公众号

由锤子便签发送 via Smartisan Notes