Python_N3_20161001_AaronYu

October 1, 2016

0.1 Summary

Pandas! Pandas! Pandas!

- 1.Data Structures
  - Series
  - DataFrame
- 2.Basic Data Exploration
  - Missing Values
  - Unique Values & Value Counts
- 3.Basic Indexing
  - position based & label based
  - iloc
  - loc
  - ix

```
In [1]: import numpy as np
        import pandas as pd
```

0.2 Data Structures

0.2.1 Series

- Seriesis a one-dimensional labeled array capable of holding any data type (integers, strings, floating point numbers, Python objects, etc.).The axis labels are collectively referred to as the index. The basic method to create a Series is to call: ser = pd.Series(data, index=idx)

```
In [11]: ser1 = pd.Series([1,2,5,7,9])
         ser1

Out[11]: 0    1
         1    2
         2    5
         3    7
         4    9
         dtype: int64
```

```
In [12]: ser2 = pd.Series(np.arange(5), index = ['a','b','c','d','e'])
         ser2

Out[12]: a    0
         b    1
         c    2
         d    3
         e    4
         dtype: int64

In [13]: dic1 = {'M':1, 'L':2,'XL':3}
         ser3 = pd.Series(dic1)
         ser3

Out[13]: L     2
         M     1
         XL    3
         dtype: int64
```

- Series is ndarray-like

```
In [14]: ser1[0]

Out[14]: 1

In [15]: ser1[:3]

Out[15]: 0    1
         1    2
         2    5
         dtype: int64

In [22]: np.sqrt(ser2)

Out[22]: a    0.000000
         b    1.000000
         c    1.414214
         d    1.732051
         e    2.000000
         dtype: float64
```

- Series is dict-like

```
In [16]: ser3['M']

Out[16]: 1

In [17]: pd.Series(['M','L']).map(ser3)

Out[17]: 0    1
         1    2
         dtype: int64
```

0.2.2 DataFrame

- DataFrame is a 2-dimensional labeled data structure with columns of potentially different types. You can think of it like a spreadsheet or SQL table. The basic method to create a DataFrame is to call:df=pd.DataFrame(data, index = idx, columns = col)

```
In [24]: df1 = pd.DataFrame(np.arange(12).reshape(6,2), columns = ['age','weight'])
         df1
```

Out[24]:

	age	weight
0	0	1
1	2	3
2	4	5
3	6	7
4	8	9
5	10	11

```
In [49]: tips = pd.read_csv('N3_tips.csv')
         tips.head()
```

Out[49]:

	total_bill	tip	sex	smoker	day	time	size
0	12.90	1.10	Female	Yes	Sat	Dinner	2.0
1	11.87	1.63	Female	No	Thur	Lunch	2.0
2	3.07	1.00	Female	Yes	Sat	Dinner	1.0
3	24.08	2.92	Female	No	Thur	Lunch	4.0
4	18.71	4.00	Male	Yes	Thur	Lunch	3.0

0.3 Basic Data Exploration

0.3.1 Missing Values

- pandas uses the floating point value NaN(Not a Number) to represent missing data in both floating as well as in non-floating point arrays. Python native None is also treated as NA

```
In [87]: tips_na = tips[10:20]
         tips_na
```

Out[87]:

	total_bill	tip	sex	smoker	day	time	size
10	18.290	3.76	Male	Yes	Sat	Dinner	4.0
11	8.770	2.00	Male	No	Sun	Dinner	2.0
12	35.260	5.00	Female	No	Sun	Dinner	4.0
13	16.595	2.00	Male	No	Sun	Dinner	2.0
14	12.460	1.50	Male	No	Fri	Dinner	2.0
15	16.820	4.00	Male	Yes	Sun	Dinner	2.0
16	16.595	NaN	Male	No	NaN	Dinner	NaN
17	14.000	3.00	Male	No	Sat	Dinner	2.0
18	10.090	2.00	Female	Yes	Fri	Lunch	2.0
19	17.070	3.00	Female	No	Sat	Dinner	3.0

- isnull() notnull()

```
In [52]: tips_na['total_bill'].isnull()

Out[52]: 10    False
         11    False
         12    False
         13     True
         14    False
         15    False
         16     True
         17    False
         18    False
         19    False
         Name: total_bill, dtype: bool

In [54]: tips_na['total_bill'].isnull().sum()

Out[54]: 2

In [55]: tips_na.isnull()

Out[55]:    total_bill    tip    sex smoker    day   time   size
         10      False  False  False  False  False  False  False
         11      False  False  False  False  False  False  False
         12      False  False  False  False  False  False  False
         13       True  False  False  False  False  False  False
         14      False  False  False  False  False  False  False
         15      False  False  False  False  False  False  False
         16       True   True  False  False   True  False   True
         17      False  False  False  False  False  False  False
         18      False  False  False  False  False  False  False
         19      False  False  False  False  False  False  False

In [56]: tips_na.isnull().sum()

Out[56]: total_bill    2
         tip           1
         sex           0
         smoker        0
         day           1
         time          0
         size          1
         dtype: int64

In [59]: tips_na.isnull().any()

Out[59]: total_bill     True
         tip            True
         sex           False
         smoker        False
```

```
day            True
time          False
size           True
dtype: bool
```

```
In [62]: tips_na.isnull().any(axis = 1)
```

```
Out[62]: 10    False
         11    False
         12    False
         13     True
         14    False
         15    False
         16     True
         17    False
         18    False
         19    False
         dtype: bool
```

```
In [63]: tips_na.isnull().any(axis = 1).sum()
```

```
Out[63]: 2
```

- dropna()

```
In [80]: tips_dropna = tips_na.dropna()
         tips_dropna
```

```
Out[80]:     total_bill   tip     sex smoker  day    time  size
         10       18.29  3.76    Male    Yes  Sat  Dinner   4.0
         11        8.77  2.00    Male     No  Sun  Dinner   2.0
         12       35.26  5.00  Female     No  Sun  Dinner   4.0
         14       12.46  1.50    Male     No  Fri  Dinner   2.0
         15       16.82  4.00    Male    Yes  Sun  Dinner   2.0
         17       14.00  3.00    Male     No  Sat  Dinner   2.0
         18       10.09  2.00  Female    Yes  Fri   Lunch   2.0
         19       17.07  3.00  Female     No  Sat  Dinner   3.0
```

- fillna()

```
In [81]: tips_fillna1 = tips_na.fillna(0)
         tips_fillna1
```

```
Out[81]:     total_bill   tip     sex smoker  day    time  size
         10       18.29  3.76    Male    Yes  Sat  Dinner   4.0
         11        8.77  2.00    Male     No  Sun  Dinner   2.0
         12       35.26  5.00  Female     No  Sun  Dinner   4.0
         13        0.00  2.00    Male     No  Sun  Dinner   2.0
         14       12.46  1.50    Male     No  Fri  Dinner   2.0
```

```
15       16.82  4.00    Male    Yes  Sun  Dinner   2.0
16        0.00  0.00    Male     No    0  Dinner   0.0
17       14.00  3.00    Male     No  Sat  Dinner   2.0
18       10.09  2.00  Female    Yes  Fri   Lunch   2.0
19       17.07  3.00  Female     No  Sat  Dinner   3.0
```

```
In [89]: tips_fillna2= tips_na.fillna({'total_bill':15, 'tip':3, 'size':2})
         tips_fillna2
```

```
Out[89]:     total_bill   tip     sex smoker  day    time  size
         10      18.290  3.76    Male    Yes  Sat  Dinner   4.0
         11       8.770  2.00    Male     No  Sun  Dinner   2.0
         12      35.260  5.00  Female     No  Sun  Dinner   4.0
         13      16.595  2.00    Male     No  Sun  Dinner   2.0
         14      12.460  1.50    Male     No  Fri  Dinner   2.0
         15      16.820  4.00    Male    Yes  Sun  Dinner   2.0
         16      16.595  3.00    Male     No  NaN  Dinner   2.0
         17      14.000  3.00    Male     No  Sat  Dinner   2.0
         18      10.090  2.00  Female    Yes  Fri   Lunch   2.0
         19      17.070  3.00  Female     No  Sat  Dinner   3.0
```

0.3.2 Unique Values & Value Counts

- unique()

```
In [94]: tips2 = tips.dropna()
```

```
In [99]: tips2['day'].unique()
```

```
Out[99]: array(['Sat', 'Thur', 'Sun', 'Fri'], dtype=object)
```

```
In [98]: len(tips2['day'].unique())
```

```
Out[98]: 4
```

- value_counts()

```
In [100]: tips2['day'].value_counts()
```

```
Out[100]: Sun     10
          Sat      8
          Thur     6
          Fri      2
          Name: day, dtype: int64
```

0.4 Basic Indexing

- indexing in pandas has two types: position based & label based

0.4.1 .iloc

- Mostly position based

```
In [102]: tips_idx = tips[:5]
          tips_idx
```

Out[102]:

	total_bill	tip	sex	smoker	day	time	size
0	12.90	1.10	Female	Yes	Sat	Dinner	2.0
1	11.87	1.63	Female	No	Thur	Lunch	2.0
2	3.07	1.00	Female	Yes	Sat	Dinner	1.0
3	24.08	2.92	Female	No	Thur	Lunch	4.0
4	18.71	4.00	Male	Yes	Thur	Lunch	3.0

```
In [104]: tips_idx.iloc[0,3]
```

```
Out[104]: 'Yes'
```

```
In [105]: tips_idx.iloc[:3, 1:]
```

Out[105]:

	tip	sex	smoker	day	time	size
0	1.10	Female	Yes	Sat	Dinner	2.0
1	1.63	Female	No	Thur	Lunch	2.0
2	1.00	Female	Yes	Sat	Dinner	1.0

0.4.2 loc

- Mostly label based

```
In [112]: tips_idx.loc[:3,['tip','sex','smoker']]
          ### note that contrary to usual python slices, BOTH the start and the sto
```

Out[112]:

	tip	sex	smoker
0	1.10	Female	Yes
1	1.63	Female	No
2	1.00	Female	Yes
3	2.92	Female	No

```
In [114]: tips_idx2 = tips_idx.set_index('day')
          tips_idx2
```

Out[114]:

day	total_bill	tip	sex	smoker	time	size
Sat	12.90	1.10	Female	Yes	Dinner	2.0
Thur	11.87	1.63	Female	No	Lunch	2.0
Sat	3.07	1.00	Female	Yes	Dinner	1.0
Thur	24.08	2.92	Female	No	Lunch	4.0
Thur	18.71	4.00	Male	Yes	Lunch	3.0

```
In [118]: tips_idx2.loc['Thur','size']
```

```
Out[118]: day
          Thur    2.0
          Thur    4.0
          Thur    3.0
          Name: size, dtype: float64

In [120]: tips_idx2.loc[['Sat'],['total_bill','size']]

Out[120]:      total_bill  size
          day
          Sat       12.90   2.0
          Sat        3.07   1.0
```

0.4.3 ix

- .ix supports mixed integer and label based access. It is the most general and will support any of the inputs in .loc and .iloc

```
In [122]: tips_idx.ix[:3, 'smoker']

Out[122]: 0    Yes
          1     No
          2    Yes
          3     No
          Name: smoker, dtype: object

In [123]: tips_idx.ix[[0,2,4], ['sex','smoker']]

Out[123]:       sex smoker
          0  Female    Yes
          2  Female    Yes
          4    Male    Yes

In [153]: tips_idx.ix[tips_idx['smoker'] == 'Yes', ['total_bill', 'tip', 'time']]

Out[153]:    total_bill  tip    time
          0       12.90  1.1  Dinner
          2        3.07  1.0  Dinner
          4       18.71  4.0   Lunch
```