Pandas notes

June 15, 2019

1 Finding entries in a dataframe that satisfy a set of given condition

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

let us consider the following dataframe:

```
In [2]: arr=np.array([[1,2,6],[3,4,7],[1,5,4]])
        df=pd.DataFrame(arr, columns=['x1','x2','x3'])

In [3]: df

Out[3]:    x1  x2  x3
        0   1   2   6
        1   3   4   7
        2   1   5   4
```

let's say we want to find the entries which have 'x1'=1

```
In [4]: df.loc[df['x1']==1]

Out[4]:    x1  x2  x3
        0   1   2   6
        2   1   5   4
```

now let us try to find the entries which have 'x1'=1 and 'x2'=2.

- As before we will use pd.DataFrame.loc to find the relevant entries
- caution 1: in order to combine multiple conditions we have to use the bitwise operators '&' and '|', as opposed to pure Python ones ('and' and 'or').
- caution 2: each condition must be enveloped in paranthesis
- Thus the syntax for searching a dataframe 'df' will be : df.loc[(condition 1) & (condition 2) ...]

```
In [5]: df.loc[(df['x1']==1) & (df['x2']==2)]

Out[5]:    x1  x2  x3
        0   1   2   6
```

2 Applying a function to the entries in a dataframe

The function to be applied can be defined explicitly or by using lambda.

We will apply the function through the pandas method called 'apply'

Suppose we want to obtain the square of entries in column 'x1' of df

```
In [6]: df['x1'].apply(lambda x: x**2)

Out[6]: 0    1
        1    9
        2    1
        Name: x1, dtype: int64
```

We will have to pass axis=1 if we want to apply a function whose arguments correspond to enteries in multiple columns of a row.

For e.g., suppose we want to add the enteries under columns 'x1' and 'x2' in each row of df

```
In [7]: res=df.apply(lambda x: x['x1']+ x['x2'], axis=1)
        print(res)

0    3
1    7
2    6
dtype: int64
```

By default, 'apply' always returns a pandas.Series

```
In [8]: type(res)

Out[8]: pandas.core.series.Series
```

let us now define a function that returns a list whose first element is the sum of entries in 'x1' and 'x2' and the second element is the difference between 'x1' and 'x2', and apply this to the rows of df

```
In [9]: def add_subtract(x1,x2):
            return [x1+x2, x1-x2]

In [10]: add_sub=df.apply(lambda x: add_subtract(x['x1'], x['x2']) ,axis=1)
         print(add_sub)

0    [3, -1]
1    [7, -1]
2    [6, -4]
dtype: object
```

Note that the result of the above code is a pandas Series made up of the list.

If we want the result to be a dataframe with the elements of the list assigned to individual columns, the we have to pass result_type = 'expand'

```
In [11]: add_sub_new=df.apply(lambda x: add_subtract(x['x1'],x['x2']) , axis=1, result_type='e
         print(add_sub_new)

   0  1
0  3 -1
1  7 -1
2  6 -4
```

add_sub_new is now a pandas Dataframe

```
In [12]: type(add_sub_new)

Out[12]: pandas.core.frame.DataFrame
```

by default, the columns of add_sub_new will be labeled by numbers: 0,1,2,...

We can rename the columns by using the method rename and passing a dictionary that maps old column names to new ones

```
In [13]: add_sub_new=add_sub_new.rename(columns={0:'sum',1:'difference'})

In [14]: print(add_sub_new)

   sum  difference
0    3          -1
1    7          -1
2    6          -4
```

3 Concatinating dataframes

This can be done using the method 'concat' in pandas

By default the dataframes will be concatenated along axis=0 i.e. vertically

To see this, let us first creat a second dataframe df2

```
In [15]: df2=pd.DataFrame(np.array([[0,1,9],[5,6,4]]), columns=['x1','x2','x3'])

In [16]: df2

Out[16]:    x1  x2  x3
         0   0   1   9
         1   5   6   4
```

now concatenate df and df2

```
In [17]: df_new=pd.concat([df,df2])
         print(df_new)
```

```
   x1  x2  x3
0   1   2   6
1   3   4   7
2   1   5   4
0   0   1   9
1   5   6   4
```

Note that here in the above code, it copied the index values as is from each dataframe. Thus we get two entries with index=0 etc. This can be fixed by passing 'ignore_index'=True

```
In [18]: df_new=pd.concat([df,df2], ignore_index=True)
         print(df_new)
```

```
   x1  x2  x3
0   1   2   6
1   3   4   7
2   1   5   4
3   0   1   9
4   5   6   4
```

In order to append two dataframe along axis=1, i.e., horizontally, we have to pass axis=1

```
In [19]: df_new2=pd.concat([df, add_sub_new], axis=1)
         print(df_new2)
```

```
   x1  x2  x3  sum  difference
0   1   2   6    3          -1
1   3   4   7    7          -1
2   1   5   4    6          -4
```

```
In [ ]:
```