3-key-operations-on-dataframes

March 10, 2023

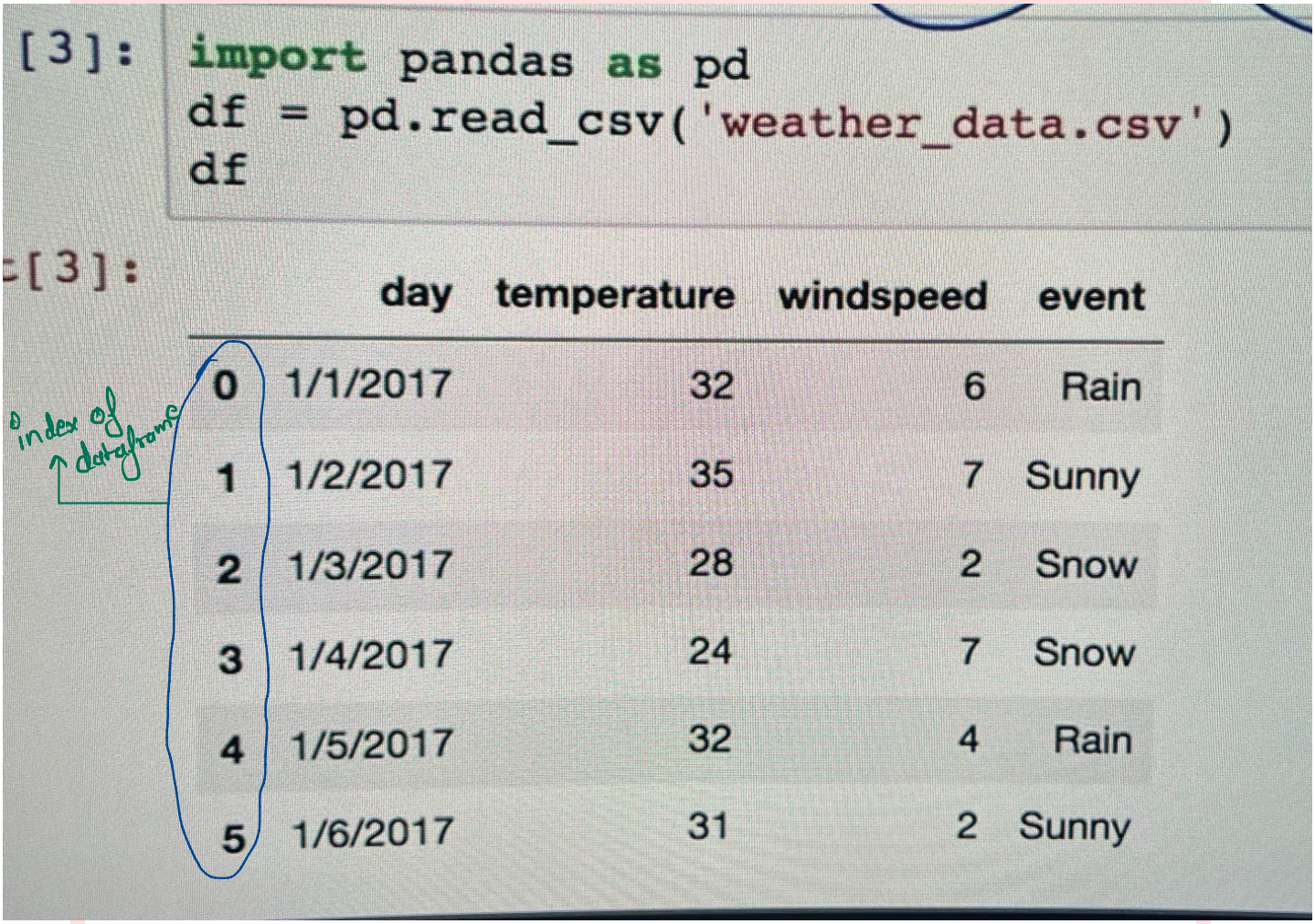
1 Read and write CSV and XLS files

Excel file (store data in table)

.xls
.xlsx

→ filename extensions

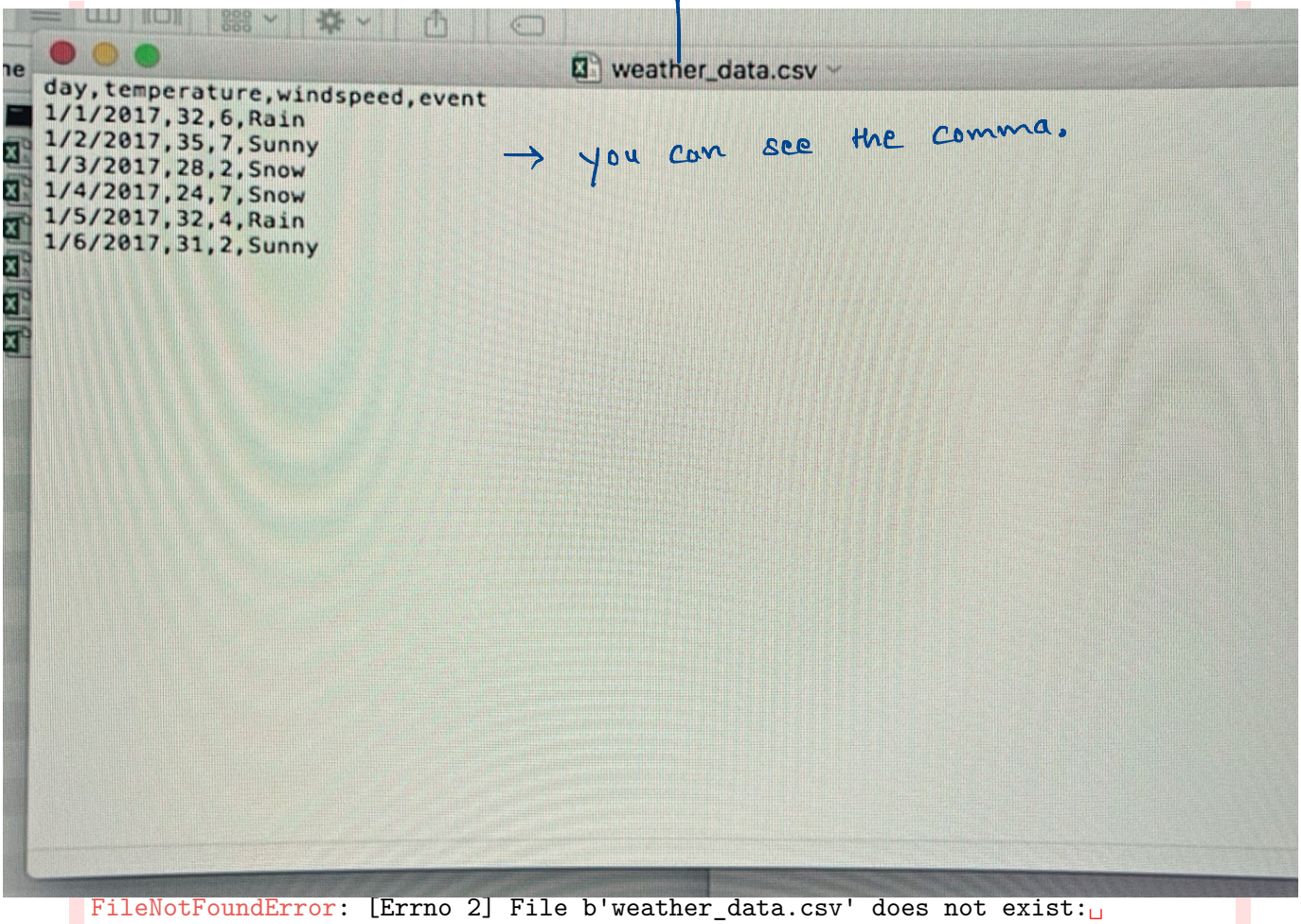
```
[ ]: import pandas as pd
     df = pd.read_csv('weather_data.csv')
     df
```



	day	temperature	windspeed	event
0	1/1/2017	32	6	Rain
1	1/2/2017	35	7	Sunny
2	1/3/2017	28	2	Snow
3	1/4/2017	24	7	Snow
4	1/5/2017	32	4	Rain
5	1/6/2017	31	2	Sunny

```
/usr/local/lib/python3.6/dist-packages/pandas/io/parsers.py in __init__(self, f
↪engine, **kwds)
    893             self.options['has_index_names'] = kwds['has_index_names']
```

This is how data store in 'csv' file.



→ you can see the comma.

```
FileNotFoundError: [Errno 2] File b'weather_data.csv' does not exist:␣
  ↪b'weather_data.csv'
```

→ package which help load excel file.

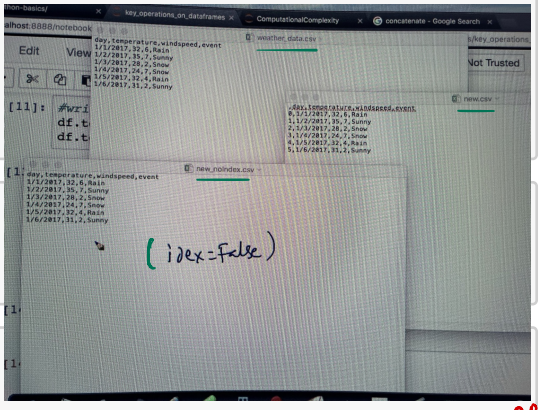
```
[ ]: #INSTALL: pip3 install xlrd

     #read excel file
     df = pd.read_excel('weather_data.xlsx')
     df
```

→ How to store 'csv' data in pc.

```
[ ]: #write DF to csv
     df.to_csv('new.csv')
     df.to_csv('new_noIndex.csv', index=False)
```

file name



if you don't want index.

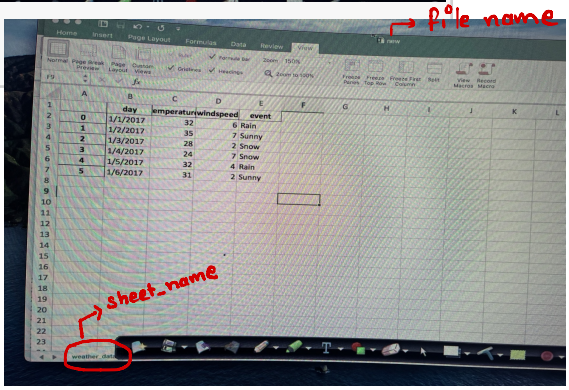
→ How to store 'xlsx' file in pc.

excel

```
[ ]: # INSTALL: pip3 install openpyxl

     #write DF to Excel
     df.to_excel('new.xlsx', sheet_name='weather_data')
```

python



single Excel file → contain → multiple sheet [a sheet → Table]

2 GROUP-BY

[ ]:
```
import pandas as pd
df = pd.read_csv('weather_data_cities.csv')
df #weather by cities
```

[ ]:

	day	city	temperature	windspeed	event
0	1/1/2017	new york	32	6	Rain
1	1/2/2017	new york	36	7	Sunny
2	1/3/2017	new york	28	12	Snow
3	1/4/2017	new york	33	7	Sunny
4	1/1/2017	mumbai	90	5	Sunny
5	1/2/2017	mumbai	85	12	Fog
6	1/3/2017	mumbai	87	15	Fog
7	1/4/2017	mumbai	92	5	Rain
8	1/1/2017	paris	45	20	Sunny
9	1/2/2017	paris	50	13	Cloudy
10	1/3/2017	paris	54	8	Cloudy
11	1/4/2017	paris	42	10	Cloudy

[ ]:
```
g = df.groupby('city')
g
```
→ what it does is, it take all of the rows where 'city' names are same and it puts them into a group.

[ ]:
```
<pandas.core.groupby.DataFrameGroupBy object at 0x106d495f8>
```

[ ]:
```
for city, city_df in g:
    print(city)
    print(city_df)
```
→ In a group I have city_df (city dataframe) corresponding to 'city'.

city name → mumbai

city dataframe →

	day	city	temperature	windspeed	event
4	1/1/2017	mumbai	90	5	Sunny
5	1/2/2017	mumbai	85	12	Fog
6	1/3/2017	mumbai	87	15	Fog
7	1/4/2017	mumbai	92	5	Rain

→ these are the new dataframe which correspond to city (mumbai). But, the indices are from the original dataframe.

new york

	day	city	temperature	windspeed	event
0	1/1/2017	new york	32	6	Rain
1	1/2/2017	new york	36	7	Sunny
2	1/3/2017	new york	28	12	Snow
3	1/4/2017	new york	33	7	Sunny

paris

	day	city	temperature	windspeed	event
8	1/1/2017	paris	45	20	Sunny
9	1/2/2017	paris	50	13	Cloudy
10	1/3/2017	paris	54	8	Cloudy
11	1/4/2017	paris	42	10	Cloudy

→ remember, when I am creating this dataframe the indices are attached.

[ ]:
```
#or to get specific group
g.get_group('new york')
```

[ ]:

	day	city	temperature	windspeed	event
0	1/1/2017	new york	32	6	Rain
1	1/2/2017	new york	36	7	Sunny
2	1/3/2017	new york	28	12	Snow
3	1/4/2017	new york	33	7	Sunny

[ ]:
```
#Find maximum temperature in each of the cities
print(g.max())
```

city	day	temperature	windspeed	event
mumbai	1/4/2017	92	15	Sunny
new york	1/4/2017	36	12	Sunny
paris	1/4/2017	54	20	Sunny

[ ]:
```
print(g.mean())
```

city	temperature	windspeed
mumbai	88.50	9.25
new york	32.25	8.00
paris	47.75	12.75

[ ]:
```
print(g.describe())
```

```
         temperature                                                    \
               count   mean       std   min    25%   50%    75%   max
city
mumbai           4.0  88.50  3.109126  85.0  86.50  88.5  90.50  92.0
new york         4.0  32.25  3.304038  28.0  31.00  32.5  33.75  36.0
paris            4.0  47.75  5.315073  42.0  44.25  47.5  51.00  54.0

         windspeed
             count   mean       std  min   25%   50%    75%   max
city
mumbai         4.0   9.25  5.057997  5.0  5.00   8.5  12.75  15.0
new york       4.0   8.00  2.708013  6.0  6.75   7.0   8.25  12.0
paris          4.0  12.75  5.251984  8.0  9.50  11.5  14.75  20.0
```

3 concatenate Data Frames

Join

dictionary {hashmap}

[ ]:
```
import pandas as pd
india_weather = pd.DataFrame({
    "city": ["mumbai","delhi","banglore"],
    "temperature": [32,45,30],
    "humidity": [80, 60, 78]
})

india_weather
```

one more way of creating dataframe.

[ ]:

	city	humidity	temperature
0	mumbai	80	32
1	delhi	60	45
2	banglore	78	30

[ ]:
```
us_weather = pd.DataFrame({
    "city": ["new york","chicago","orlando"],
    "temperature": [21,14,35],
    "humidity": [68, 65, 75]
})
us_weather
```

[ ]:

	city	humidity	temperature
0	new york	68	21
1	chicago	65	14
2	orlando	75	35

[ ]:
```
#concate two dataframes
df = pd.concat([india_weather, us_weather])
df
```

[ ]:

	city	humidity	temperature
0	mumbai	80	32
1	delhi	60	45
2	banglore	78	30
0	new york	68	21
1	chicago	65	14
2	orlando	75	35

Just copying the indices.

[ ]:
```
#if you want continuous index
df = pd.concat([india_weather, us_weather], ignore_index=True)
df
```

[ ]:

	city	humidity	temperature
0	mumbai	80	32
1	delhi	60	45

```
2  banglore        78           30
3  new york        68           21
4   chicago        65           14
5   orlando        75           35
```

[ ]:
```
df = pd.concat([india_weather, us_weather],axis=1)
df
```

[ ]:
```
       city  humidity  temperature      city  humidity  temperature
0    mumbai        80           32  new york        68           21
1     delhi        60           45   chicago        65           14
2  banglore        78           30   orlando        75           35
```

4 Merge DataFrames

[ ]:
```
temperature_df = pd.DataFrame({
    "city": ["mumbai","delhi","banglore", 'hyderabad'],
    "temperature": [32,45,30,40]})
temperature_df
```

[ ]:
```
       city  temperature
0    mumbai           32
1     delhi           45
2   banglore           30
3  hyderabad           40
```

[ ]:
```
humidity_df = pd.DataFrame({
    "city": ["delhi","mumbai","banglore"],
    "humidity": [68, 65, 75]})
humidity_df
```

[ ]:
```
       city  humidity
0     delhi        68
1    mumbai        65
2  banglore        75
```

[ ]:
```
#merge two dataframes with out explicitly mention index
df = pd.merge(temperature_df, humidity_df, on='city')
df
```

↳ I want to merge it based on 'city'.

[ ]:
```
       city  temperature  humidity
0    mumbai           32        65
1     delhi           45        68
2  banglore           30        75
```

← No hyderabad.

→ here I'm joing two table using city name.

```
[ ]: #OUTER-JOIN
df = pd.merge(temperature_df, humidity_df, on='city', how='outer')
df
```

```
[ ]:        city  temperature  humidity
0     mumbai           32      65.0
1      delhi           45      68.0
2   banglore           30      75.0
3  hyderabad           40       NaN
```

5 Numerical Indexing (.loc vs iloc)

.loc → location
.iloc → index location

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

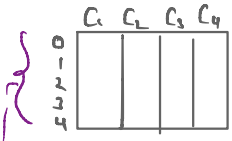
```
[ ]: df = pd.DataFrame([1,2,3,4,5,6,7,8,9,19], index=[49,48,47,46,45, 1, 2, 3, 4, 5])
df
```

index

```
[ ]:     0
49   1
48   2
47   3
46   4
45   5
1    6
2    7
3    8
4    9
5   19
```

row '0' → 49
row '1' → 48
row '2' → 47
row '3' → 46

→ whenever, we create a dataframe it also create a 'index' column.



→ by default the row no. is default index. But if you want to use your own index, use above command.

→ if I want to get a value (4) from df (dataframe)?
  → df → index 46 → s.loc[46] → I'm using index to get data.
  → df → row '3' → s.iloc[3] → " " row no. " " ".

```
[ ]: s.loc[:2]
```

49 1
48 2
⋮ ⋮
2 7

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-3-a7a6a418f874> in <module>()
----> 1 s.loc[:2]

NameError: name 's' is not defined
```

```
[ ]: s.iloc[:2]
```

```
[ ]: 49    1
48    2
dtype: int64
```

[ ]:
```
s.loc[45]
```

[ ]: 5

[ ]:
```
s.iloc[45]
```

↳ error.

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-20-a6772688a529> in <module>()
----> 1 s.iloc[45]

/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-packages/
 ↪pandas/core/indexing.py in __getitem__(self, key)
   1326         else:
   1327             key = com._apply_if_callable(key, self.obj)
-> 1328             return self._getitem_axis(key, axis=0)
   1329
   1330     def _is_scalar_access(self, key):

/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-packages/
 ↪pandas/core/indexing.py in _getitem_axis(self, key, axis)
   1747
   1748             # validate the location
-> 1749             self._is_valid_integer(key, axis)
   1750
   1751             return self._get_loc(key, axis=axis)

/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-packages/
 ↪pandas/core/indexing.py in _is_valid_integer(self, key, axis)
   1636         l = len(ax)
   1637         if key >= l or key < -l:
-> 1638             raise IndexError("single positional indexer is␣
 ↪out-of-bounds")
   1639         return True
   1640

IndexError: single positional indexer is out-of-bounds
```

[ ]: