

# **Python Training Follow-up Session**



# Pandas Important Commands

- Read/Write CSV
  - `pd.read_csv('file.csv', header=None, nrows=5)`
  - `pd.to_csv('myDataFrame.csv')`
- Read and Write to Excel
  - `>>> pd.read_excel('file.xlsx')`
  - `>>> pd.to_excel('dir/myDataFrame.xlsx', sheet_name='Sheet1')`
- Selecting Elements - `df[1:]`
- By position - `df.iloc([0], [0])`
- By Label - `df.loc([0], ['Country'])`
- Select single row of subset of rows
  - `df.ix[2]`
- Select a single column of subset of columns
  - `df.ix[:, 'Capital']`

# Pandas Important Commands (Cont...)

- Use of Filter - `df[df['Population']>1200000000]` -
- Drop values from columns(axis=1) - `df.drop('Country', axis=1)`
- Sort
  - `df.sort_index()`
  - `df.sort_values(by='Country')`
- Ranking
  - `df.rank()`
- All sort of information
  - `df.info()`
  - `Df.index`
  - `Df.columns`
  - `df.count()`
- Summary
  - `Df.sum()`
  - `df.cumsum()`
  - `df.min(), df.max()`
  - Index min/max - `df.idxmin(), df.idxmax ()`
  - `df.median()`
  - `Df.mean(), df.max()`

# Pandas Important Commands (Cont...)

- You can also apply function to elements of DataFrame
  - `df.apply(f)`
  - `df.applymap(f)`

# Reading Files - in Numpy

```
$ more array_example.txt  
1.1, 1.2, 1.3, 1.4  
2.1, 2.2, 2.3, 2.4  
3.1, 3.2, 3.3, 3.4  
4.1, 4.2, 4.3, 4.4  
$
```

```
(>>> arr = np.loadtxt('array_example.txt', delimiter=',')  
>>> arr  
array([[ 1.1,  1.2,  1.3,  1.4],  
       [ 2.1,  2.2,  2.3,  2.4],  
       [ 3.1,  3.2,  3.3,  3.4],  
       [ 4.1,  4.2,  4.3,  4.4]])  
>>>
```

Numpy Module can read any text file - based on any “delimiter”. In the above example “array\_example.txt” with delimiter produces new Numpy array with The help of “np.loadtxt” method.

# Generate Data Files: In Pandas

```
>>> import pandas as pd
>>> twiki = pd.read_csv('TWiki_Application.log', delimiter='|')
>>> twiki.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6911 entries, 0 to 6910
Data columns (total 8 columns):
Unnamed: 0          0 non-null float64
 2017-01-01 - 07:01:43  6911 non-null object
JawaharlalNehru     6911 non-null object
view                6911 non-null object
Main.WebHome        6911 non-null object
Mozilla             6911 non-null object
253.111.182.73      6911 non-null object
                   6911 non-null object
dtypes: float64(1), object(7)
memory usage: 432.0+ KB
```

# Generate Data Files: In Pandas

```
>>> twiki.columns=['junk1', 'timestamp', 'user', 'action', 'topic', 'browser', 'IP', 'junk2']
>>> twiki.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6911 entries, 0 to 6910
Data columns (total 8 columns):
junk1          0 non-null float64
timestamp      6911 non-null object
user           6911 non-null object
action         6911 non-null object
topic          6911 non-null object
browser        6911 non-null object
IP             6911 non-null object
junk2          6911 non-null object
dtypes: float64(1), object(7)
memory usage: 432.0+ KB

>>> twiki = twiki.drop('junk1', 1)
>>> twiki = twiki.drop('junk2', 1)
>>> twiki.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6911 entries, 0 to 6910
Data columns (total 6 columns):
timestamp      6911 non-null object
user           6911 non-null object
action         6911 non-null object
topic          6911 non-null object
browser        6911 non-null object
IP             6911 non-null object
dtypes: object(6)
memory usage: 324.0+ KB

>>> user = twiki.groupby('user')
>>> type(user)
<class 'pandas.core.groupby.DataFrameGroupBy'>
>>>
```

# Generate Data Files: In Pandas

```
>>> user = twiki.groupby('user')
>>> type(user)
<class 'pandas.core.groupby.DataFrameGroupBy'>
>>> user.count()
```

|                | timestamp | action | topic | browser | IP  |
|----------------|-----------|--------|-------|---------|-----|
| user           |           |        |       |         |     |
| AbrahamLincoln | 151       | 151    | 151   | 151     | 151 |
| AlbertEinstein | 142       | 142    | 142   | 142     | 142 |
| AungsansuuKyi  | 147       | 147    | 147   | 147     | 147 |
| BarackObama    | 136       | 136    | 136   | 136     | 136 |
| BenazirBhutto  | 119       | 119    | 119   | 119     | 119 |
| BillGates      | 151       | 151    | 151   | 151     | 151 |
| CharlesDemings | 146       | 146    | 146   | 146     | 146 |



# Generate Data Files: In Pandas

```
>>> import pandas as pd
>>>
>>>
>>> xls_file = pd.ExcelFile('example.xls')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/usr/local/lib/python3.5/dist-packages/pandas/io/excel.py", line 220, in __init__
    import xlrd # throw an ImportError if we need to
ImportError: No module named 'xlrd'
>>>
$ pip2 install xlrd
bash: pip2: command not found
$ pip3 install xlrd
Collecting xlrd
  Downloading xlrd-1.0.0-py3-none-any.whl (143kB)
    100% |#####| 153kB 34kB/s
Installing collected packages: xlrd
Successfully installed xlrd-1.0.0
```

# Generate Data Files: In Pandas

```
>>> import pandas as pd
>>> xls_file = pd.ExcelFile('example.xls')
>>> xls_file
<pandas.io.excel.ExcelFile object at 0x7fddddd554710>
>>> xls_file.sheet_names
['Sheet1']
>>> df = xls_file.parse('Sheet1')
>>> df
```

|   | year | deaths_attacker | deaths_defender | soldiers_attacker \ |
|---|------|-----------------|-----------------|---------------------|
| 0 | 1945 | 425             | 423             | 2532                |
| 1 | 1956 | 242             | 264             | 6346                |
| 2 | 1964 | 323             | 1231            | 3341                |
| 3 | 1969 | 223             | 23              | 6732                |
| 4 | 1971 | 783             | 23              | 12563               |
| 5 | 1981 | 436             | 42              | 2356                |
| 6 | 1982 | 324             | 124             | 253                 |
| 7 | 1992 | 3321            | 631             | 5277                |
| 8 | 1999 | 262             | 232             | 2732                |
| 9 | 2004 | 843             | 213             | 6278                |

  

|   | soldiers_defender | wounded_attacker | wounded_defender |
|---|-------------------|------------------|------------------|
| 0 | 37235             | 41               | 14               |
| 1 | 2523              | 214              | 1424             |
| 2 | 2133              | 131              | 131              |
| 3 | 1245              | 12               | 12               |
| 4 | 2671              | 123              | 34               |
| 5 | 7832              | 124              | 124              |
| 6 | 2622              | 264              | 1124             |
| 7 | 3331              | 311              | 1431             |
| 8 | 2522              | 132              | 122              |
| 9 | 26773             | 623              | 2563             |

```
>>>
```

# Generate Data Files: In Pandas

```
>>> from pandas import ExcelWriter
>>> import pandas as pd
>>> xls_file = pd.ExcelFile('example.xls')
>>> df = xls_file.parse('Sheet1')
>>> df
```

|   | year | deaths_attacker | deaths_defender | soldiers_attacker | \ |
|---|------|-----------------|-----------------|-------------------|---|
| 0 | 1945 | 425             | 423             | 2532              |   |
| 1 | 1956 | 242             | 264             | 6346              |   |
| 2 | 1964 | 323             | 1231            | 3341              |   |
| 3 | 1969 | 223             | 23              | 6732              |   |
| 4 | 1971 | 783             | 23              | 12563             |   |
| 5 | 1981 | 436             | 42              | 2356              |   |
| 6 | 1982 | 324             | 124             | 253               |   |
| 7 | 1992 | 3321            | 631             | 5277              |   |
| 8 | 1999 | 262             | 232             | 2732              |   |
| 9 | 2004 | 843             | 213             | 6278              |   |

  

|   | soldiers_defender | wounded_attacker | wounded_defender |
|---|-------------------|------------------|------------------|
| 0 | 37235             | 41               | 14               |
| 1 | 2523              | 214              | 1424             |
| 2 | 2133              | 131              | 131              |
| 3 | 1245              | 12               | 12               |
| 4 | 2671              | 123              | 34               |
| 5 | 7832              | 124              | 124              |
| 6 | 2622              | 264              | 1124             |
| 7 | 3331              | 311              | 1431             |
| 8 | 2522              | 132              | 122              |
| 9 | 26773             | 623              | 2563             |

```
>>> writer = ExcelWriter('new_example.xlsx')
>>> df.to_excel(writer, 'Sheet1')
>>> writer.save()
```

# Generate Data Files - In pandas

write\_csv.py

```
#!/usr/bin/python3
import pandas as pd
data = {
    'year': ['2002', '2003', '2004', '2005'],
    'speed': ['9.0', '10.0', '9.5', '8.9'],
}
df = pd.DataFrame(data)
print (df)

#df.to_csv('myspeed.csv', sep='|', index_label='Number')
df.to_csv('myspeed.csv', sep='|', index=False)
```

# Filtering

**`filter_dataframes.py`**

# Playing with timestamps

[twiki\\_log\\_analyse.py](#)

- reading log file - using delimiter as “|”
- Converting string date Data to “datetime” object
- Selecting data for particular date range

# Reading SAS File

```
$ python3
Python 3.5.2 (default, Nov 17 2016, 17:05:23)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import pandas as pd
>>> df = pd.read_sas('test1.sas7bdat')
>>> df.head(1)
   Column1  Column2  Column3  Column4  Column5  Column6  Column7  Column8  \
0    0.636  b'pear'    84.0  1965-12-10    0.103  b'apple'    20.0    NaN

   Column9  Column10  ...  Column91  Column92  Column93  Column94  \
0    0.621  b'apple'  ...    87.0    5587.0    0.94  b'apple'

   Column95  Column96  Column97  Column98  Column99  Column100
0    50.0    1611.0    0.48  b'apple'    45.0    3230.0

[1 rows x 100 columns]
>>>
```



# Merge - Columns

df\_a

|   | subject_id | first_name | last_name |
|---|------------|------------|-----------|
| 0 | 1          | Alex       | Anderson  |
| 1 | 2          | Amy        | Ackerman  |
| 2 | 3          | Allen      | Ali       |
| 3 | 4          | Alice      | Aoni      |
| 4 | 5          | Ayoung     | Atiches   |

df\_b

|   | subject_id | first_name | last_name |
|---|------------|------------|-----------|
| 0 | 4          | Billy      | Bonder    |
| 1 | 5          | Brian      | Black     |
| 2 | 6          | Bran       | Balwner   |
| 3 | 7          | Bryce      | Brice     |
| 4 | 8          | Betty      | Btisan    |

`pd.merge(df_a, df_b, on=['subject_id', 'first_name'], how='outer')`

|   | subject_id | first_name_x | last_name_x | first_name_y | last_name_y |
|---|------------|--------------|-------------|--------------|-------------|
| 0 | 1          | Alex         | Anderson    | NaN          | NaN         |
| 1 | 2          | Amy          | Ackerman    | NaN          | NaN         |
| 2 | 3          | Allen        | Ali         | NaN          | NaN         |
| 3 | 4          | Alice        | Aoni        | Billy        | Bonder      |
| 4 | 5          | Ayoung       | Atiches     | Brian        | Black       |
| 5 | 6          | NaN          | NaN         | Bran         | Balwner     |
| 6 | 7          | NaN          | NaN         | Bryce        | Brice       |
| 7 | 8          | NaN          | NaN         | Betty        | Btisan      |



# Merge - Multiple Columns

df\_a

|   | subject_id | first_name | last_name |
|---|------------|------------|-----------|
| 0 | 1          | Alex       | Anderson  |
| 1 | 2          | Amy        | Ackerman  |
| 2 | 3          | Allen      | Ali       |
| 3 | 4          | Alice      | Aoni      |
| 4 | 5          | Ayoung     | Atiches   |

df\_b

|   | subject_id | first_name | last_name |
|---|------------|------------|-----------|
| 0 | 4          | Billy      | Bonder    |
| 1 | 5          | Brian      | Black     |
| 2 | 6          | Bran       | Balwner   |
| 3 | 7          | Bryce      | Brice     |
| 4 | 8          | Betty      | Btisan    |

`pd.merge(df_a, df_b, on=['subject_id', 'first_name'], how='outer')`

|   | subject_id | first_name | last_name_x | last_name_y |
|---|------------|------------|-------------|-------------|
| 0 | 1          | Alex       | Anderson    | NaN         |
| 1 | 2          | Amy        | Ackerman    | NaN         |
| 2 | 3          | Allen      | Ali         | NaN         |
| 3 | 4          | Alice      | Aoni        | NaN         |
| 4 | 5          | Ayoung     | Atiches     | NaN         |
| 5 | 4          | Billy      | NaN         | Bonder      |
| 6 | 5          | Brian      | NaN         | Black       |
| 7 | 6          | Bran       | NaN         | Balwner     |
| 8 | 7          | Bryce      | NaN         | Brice       |
| 9 | 8          | Betty      | NaN         | Btisan      |

[merge\\_multiple.py](#)

You need to pass sequence (Python List, Tuple) to “on” parameter

# GroupBy

## grouped\_one.py

```
import numpy as np
import pandas as pd

df = pd.DataFrame(
    {'key1' : ['a', 'a', 'b', 'b', 'a'],
     'key2' : ['one', 'two', 'one', 'two', 'one'],
     'data1' : np.random.randn(5),
     'data2' : np.random.randn(5),
    })

print(df)
grouped = df['data1'].groupby(df['key1'])
print (grouped)
print ("The mean is: ")
print (grouped.mean())
```

```
$ ./grouped_one.py
      data1      data2 key1 key2
0 -1.283124  0.674424    a  one
1  0.143696 -1.586856    a  two
2  0.690283 -1.157718    b  one
3 -0.553324 -0.037819    b  two
4 -0.508766 -1.294618    a  one
<pandas.core.groupby.SeriesGroupBy object
The mean is:
key1
a   -0.549398
b    0.068479
Name: data1, dtype: float64
$
```

# GroupBy (Cont ...)

[grouped\\_two.py](#)

```
means = df['data1'].groupby([df['key1'], df['key2']]).mean()
print (means)
```

```
key1  key2
a     one   -1.763688
      two    0.359356
b     one    0.618805
      two    1.185997
Name: data1, dtype: float64
```

# GroupBy (Cont ...)

## grouped\_three.py

```
import numpy as np
import pandas as pd

df = pd.DataFrame(
    {'key1' : ['a', 'a', 'b', 'b', 'a'],
     'key2' : ['one', 'two', 'one', 'two', 'one'],
     'data1' : np.random.randn(5),
     'data2' : np.random.randn(5),
    })
```

```
print(df)
```

```
states = np.array(['Ohio', 'California', 'California', 'Ohio', 'Ohio'])
years = np.array([2005, 2005, 2006, 2005, 2006])
```

```
print (df['data1'].groupby([states, years]).mean())
```

```
California  2005    -0.116199
            2006   -0.875065
Ohio        2005    0.482127
            2006    0.021921
Name: data1, dtype: float64
...
```

# GroupBy (Cont ...)

```
>>> import numpy as np
>>> import pandas as pd
>>>
>>> df = pd.DataFrame(
...     {'key1' : ['a', 'a', 'b', 'b', 'a'],
...      'key2' : ['one', 'two', 'one', 'two', 'one'],
...      'data1' : np.random.randn(5),
...      'data2' : np.random.randn(5),
...     })
>>>
>>> print(df)
   data1    data2 key1 key2
0 -1.189004  1.358986   a  one
1 -1.412232 -0.568271   a  two
2  1.056801  0.014768   b  one
3 -1.917031 -0.952388   b  two
4  0.149286  2.111735   a  one
>>> df.groupby('key1').mean()
      data1    data2
key1
a    -0.817316  0.967483
b    -0.430115 -0.468810
>>> df.groupby(['key1', 'key2']).mean()
      data1    data2
key1 key2
a    one -0.519859  1.735360
     two -1.412232 -0.568271
b    one  1.056801  0.014768
     two -1.917031 -0.952388
```

# GroupBy (Cont ...)

## grouped\_four.py

```
#!/usr/bin/python3

import numpy as np
import pandas as pd

df = pd.DataFrame(
    {'Col1' : ['A', 'B', 'A', 'A', 'B', 'C', 'C', 'C'],
     'Value' : [1, 2, 2, 1, 2, 3, 2, 2]},
    )

print ('df')
print(df)
print ("-----")
df2 = df.groupby(['Col1', 'Value'])
print ("df2 = df.groupby(['Col1', 'Value'])")
print (df2)

print ('df2.size()')
print (df2.size())

print ("-----")

df3 = df2.size().reset_index()
print ('df3 = df2.size().reset_index()')
print ('df3')
print (df3)
print ("-----")
df3.columns = ['Col1', 'Value', 'Count']

print(df3)
```

```
df2 = df.groupby(['Col1', 'Value'])
<pandas.core.groupby.DataFrameGroupBy object>
df2.size()
Col1  Value
A      1      2
      2      1
B      2      2
C      2      2
      3      1
dtype: int64
-----
df3 = df2.size().reset_index()
df3
   Col1  Value  Count
0     A      1      2
1     A      2      1
2     B      2      2
3     C      2      2
4     C      3      1
-----
   Col1  Value  Count
0     A      1      2
1     A      2      1
2     B      2      2
3     C      2      2
4     C      3      1
```



# Iterating Over Groups

## iterating\_one.py

```
import numpy as np
import pandas as pd

df = pd.DataFrame(
    {'key1' : ['a', 'a', 'b', 'b', 'a'],
     'key2' : ['one', 'two', 'one', 'two', 'one'],
     'data1' : np.random.randn(5),
     'data2' : np.random.randn(5),
    })

print(df)

for name, group in df.groupby('key1'):
    print(name, "---->")
    print(group)
```

```
($ ./iterating_one.py
      data1      data2 key1 key2
0 -0.205060 -2.382546    a  one
1 -0.145556 -0.777704    a  two
2 -0.546452 -0.405837    b  one
3 -0.047827 -1.294755    b  two
4 -1.127878 -1.443541    a  one
a ---->
      data1      data2 key1 key2
0 -0.205060 -2.382546    a  one
1 -0.145556 -0.777704    a  two
4 -1.127878 -1.443541    a  one
b ---->
      data1      data2 key1 key2
2 -0.546452 -0.405837    b  one
3 -0.047827 -1.294755    b  two
```

# Iterating Over Groups

## iterating\_multiple.py

```
import numpy as np
import pandas as pd

df = pd.DataFrame(
    {'key1' : ['a', 'a', 'b', 'b', 'a'],
     'key2' : ['one', 'two', 'one', 'two', 'one'],
     'data1' : np.random.randn(5),
     'data2' : np.random.randn(5),
    })

print(df)

for (k1, k2), group in df.groupby(['key1', 'key2']):
    print(k1, k2, "---->")
    print(group)
```

```
$ ./iterating_multiple.py
      data1      data2 key1 key2
0 -0.694805  0.683633    a  one
1  0.905778 -0.494622    a  two
2 -0.569520 -0.679289    b  one
3  0.422125  0.471756    b  two
4  0.483324  0.213555    a  one
a one ---->
      data1      data2 key1 key2
0 -0.694805  0.683633    a  one
4  0.483324  0.213555    a  one
a two ---->
      data1      data2 key1 key2
1  0.905778 -0.494622    a  two
b one ---->
      data1      data2 key1 key2
2 -0.569520 -0.679289    b  one
b two ---->
      data1      data2 key1 key2
3  0.422125  0.471756    b  two
```



# Iterating - Create Dictionaries

## iterating\_dict.py

```
import numpy as np
import pandas as pd

df = pd.DataFrame(
    {'key1' : ['a', 'a', 'b', 'b', 'a'],
     'key2' : ['one', 'two', 'one', 'two', 'one'],
     'data1' : np.random.randn(5),
     'data2' : np.random.randn(5),
    })

print(df)

dict_data = dict(list(df.groupby('key1')))
#print(dict_data)

print (dict_data['a'])
```

```
$ ./iterating_dict.py
      data1      data2 key1 key2
0  0.235055  0.914145    a  one
1  0.937847  0.686546    a  two
2 -1.210439 -0.395270    b  one
3  1.388625  0.299382    b  two
4  0.580331 -0.589879    a  one
      data1      data2 key1 key2
0  0.235055  0.914145    a  one
1  0.937847  0.686546    a  two
4  0.580331 -0.589879    a  one
```

# Iterating

```
-3 df.groupby('key1')['data1']  
-3 df.groupby('key1')[['data2']]  
-3  
-3 OR  
-3  
-3 df['data1'].groupby(df['key1'])  
-3 df[['data2']].groupby(df['key1'])  
-3
```

# Iterating - Functions

## iterating\_functions.py

```
import numpy as np
import pandas as pd

df = pd.DataFrame(
    {'key1' : ['a', 'a', 'b', 'b', 'a'],
     'key2' : ['one', 'two', 'one', 'two', 'one'],
     'data1' : np.random.randn(5),
     'data2' : np.random.randn(5),
    })

print(df)

m = df.groupby(['key1', 'key2'])[['data2']].mean()
print (m)
```

```
$ ./iterating_function.py
      data1      data2 key1 key2
0  0.188186  1.040266    a  one
1  0.588322 -1.068707    a  two
2 -0.925718  0.215268    b  one
3 -1.809745  2.117613    b  two
4  0.678735 -0.729690    a  one

      data2
key1 key2
a     one  0.155288
     two -1.068707
b     one  0.215268
     two  2.117613
```

# Binning - by Pandas “cut” function

`data_binning.py`

# Pivot Table

- [pandas\\_pivot\\_one.py](#)
- [pandas\\_pivot\\_two.py](#)

# Database to Pandas

`select_task.py` (Tasks Table entries will be fetched as DataFrame)

# Sorting, remove Duplicates

[Sorting\\_multiple.py](#)

[remove\\_duplicates.py](#)

**Aegis**

SCHOOL OF BUSINESS  
SCHOOL OF DATA SCIENCE  
SCHOOL OF TELECOMMUNICATION

# If\_else - handling

[if\\_else\\_for\\_loop.py](#)

[add\\_sum\\_max\\_columns.py](#)



# Ranking Result - On Columns

[ranking\\_columns.py](#)

# sampling

stratified\_sampling.py

# Thank you

Email - [info@dravate.com](mailto:info@dravate.com)  
(Anything Related Python Programming)

