Pandas Basics

Pandas

- Open-source Python Library
- High-performance data manipulation and analysis tool
- Powerful data-structures
- Can accomplish
  - Load, prepare, manipulate, model, analyze

Features of pandas

- Fast and efficient DataFrame object.
- Data loading tools supporting different file formats.
- Data alignment and reshaping functions
- Handling missing data.
- Label-based slicing, indexing and subsetting of large data
- Columns from a data structure can be deleted or inserted.
- Group by data for aggregation and transformations.
- High performance merging and joining of data.
- Time Series functionality.

Data Structures

- Series
  - 1-D Homogenous array
  - Size immutable
  - Values mutable
- DataFrame
- Panel

Data Structures

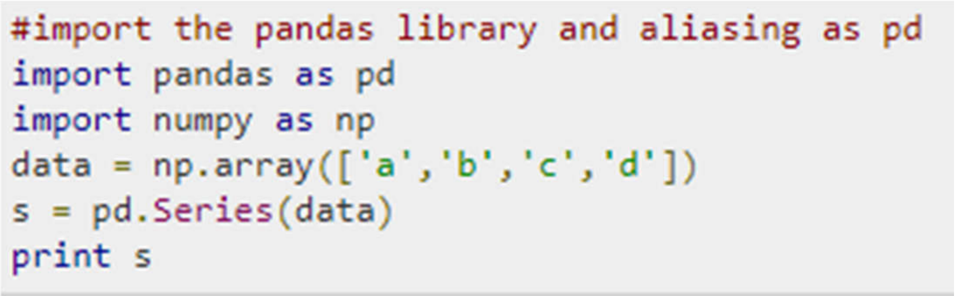
- Series
- DataFrame
  - 2-D potentially heterogeneous tabular structure
  - Size Mutable
  - Data Mutable
  - Rows and columns can have different datatype
- Panel
  - 3-D array
  - Size and data mutable

Series

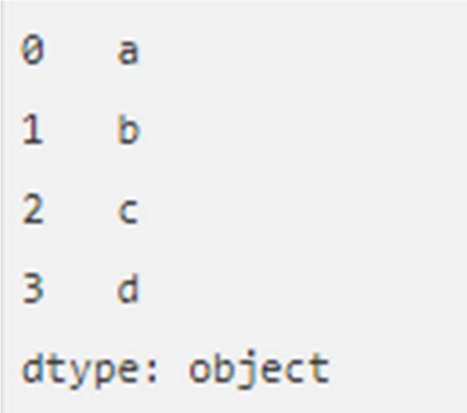
- Create a series
  - `pandas.Series( data, index, dtype, copy)`
- Data can be
  - Array
  - Dict
  - Scalar

Series

- Create a Series



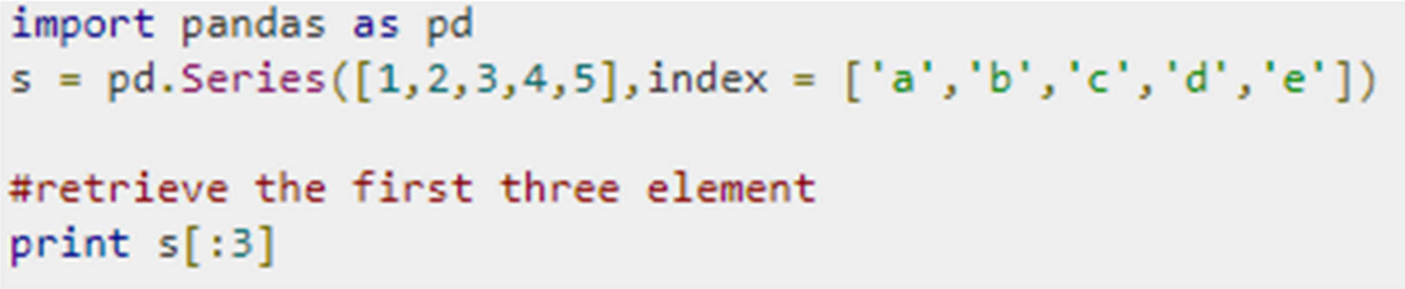
```
#import the pandas library and aliasing as pd
import pandas as pd
import numpy as np
data = np.array(['a','b','c','d'])
s = pd.Series(data)
print s
```



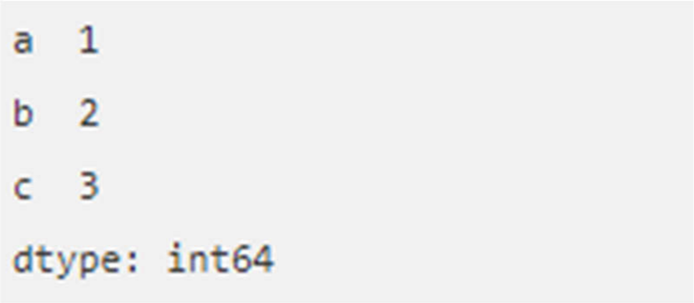
```
0    a

1    b

2    c

3    d

dtype: object
```

Series

- Access data from a Series



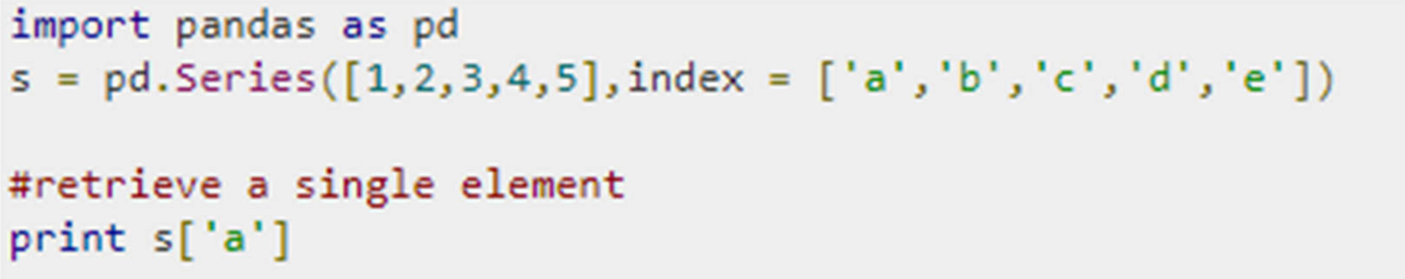
```
import pandas as pd
s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])

#retrieve the first three element
print s[:3]
```



```
a  1
b  2
c  3
dtype: int64
```

Series

- Access data from a Series



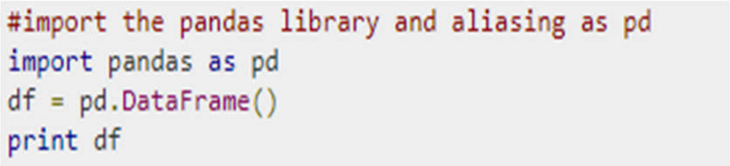
```
import pandas as pd
s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])

#retrieve a single element
print s['a']
```

DataFrame

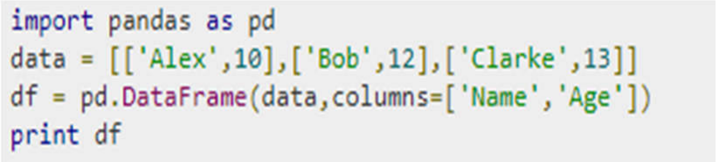
- Create a DataFrame
  - `pandas.DataFrame( data, index, columns, dtype, copy)`
- Data can be
  - Lists
  - Dict
  - Series
  - Numpy ndarray
  - DataFrame

DataFrame

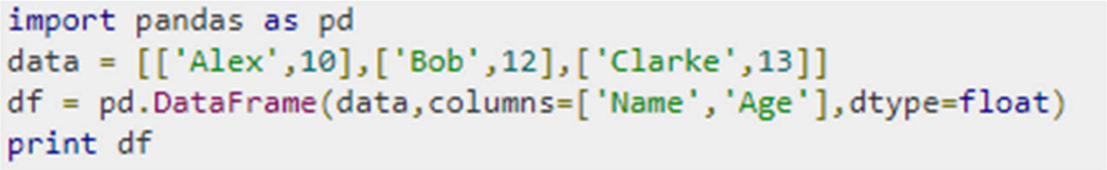
- Create a DataFrame



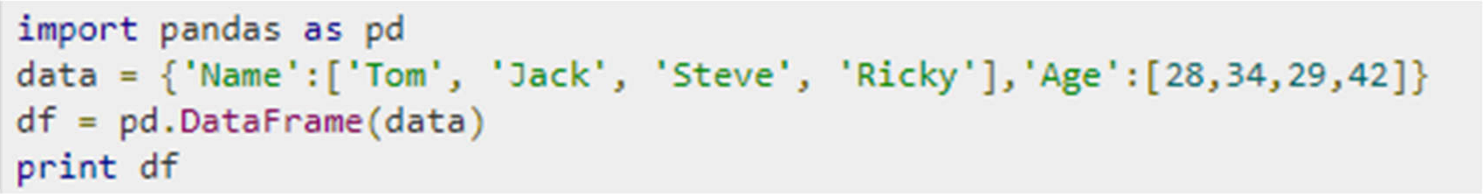
```
#import the pandas library and aliasing as pd
import pandas as pd
df = pd.DataFrame()
print df
```



```
import pandas as pd
data = [['Alex',10],['Bob',12],['Clarke',13]]
df = pd.DataFrame(data,columns=['Name','Age'])
print df
```



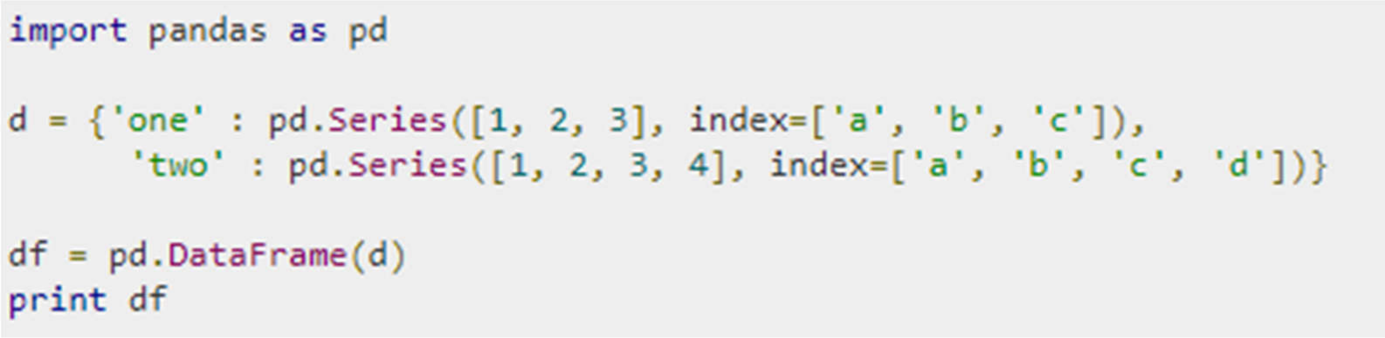
```
import pandas as pd
data = [['Alex',10],['Bob',12],['Clarke',13]]
df = pd.DataFrame(data,columns=['Name','Age'],dtype=float)
print df
```



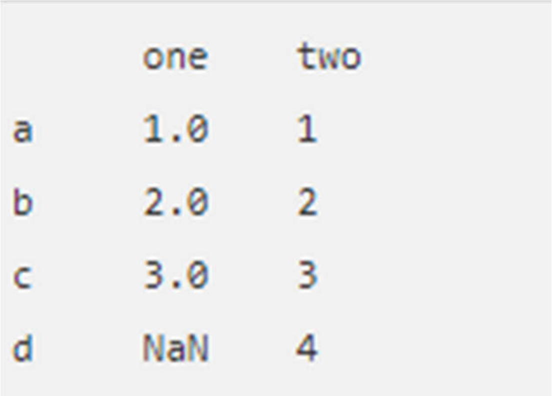
```
import pandas as pd
data = {'Name':['Tom', 'Jack', 'Steve', 'Ricky'],'Age':[28,34,29,42]}
df = pd.DataFrame(data)
print df
```

DataFrame

- Create a DataFrame



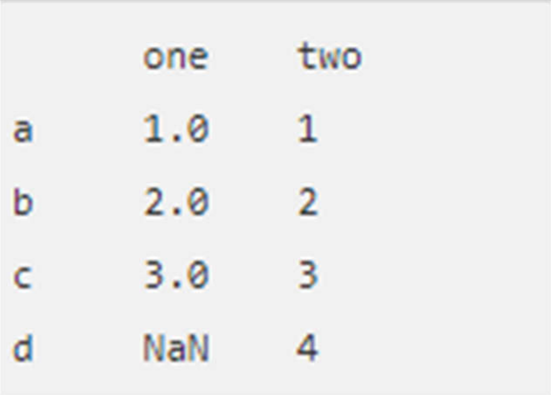
```python
import pandas as pd

d = {'one' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),
      'two' : pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}

df = pd.DataFrame(d)
print df
```



```
      one    two
a     1.0    1
b     2.0    2
c     3.0    3
d     NaN    4
```

DataFrame

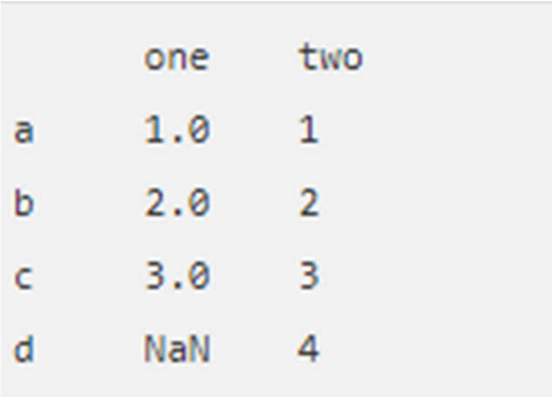
- Column Accessing
  - `df[‘one’]`
- Column Addition
  - `df[‘three’]=df[‘one’]+df[‘two’]`
- Column Deletion
  - `del df[‘one’]`



	one	two
a	1.0	1
b	2.0	2
c	3.0	3
d	NaN	4

DataFrame

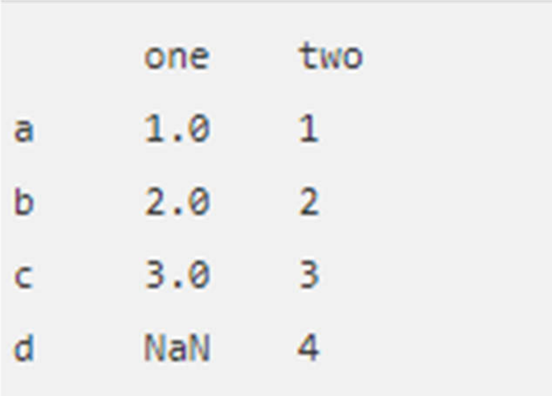
- Row Selection
  - By label
    - `df.loc[‘b']`
  - By integer location:
    - `df.iloc[2]`
  - By slicing
    - `df[2:4]`



	one	two
a	1.0	1
b	2.0	2
c	3.0	3
d	NaN	4

DataFrame

- Row Addition
  - `df.append()`
- Row Deletion
  - `df.drop()`



```
     one    two
a    1.0    1
b    2.0    2
c    3.0    3
d    NaN    4
```

Basic DataFrame Functionality

S.No.	Attribute or Method	Description
1	T	Transposes rows and columns.
2	axes	Returns a list with the row axis labels and column axis labels as the only members.
3	dtypes	Returns the dtypes in this object.
4	empty	True if NDFrame is entirely empty [no items]; if any of the axes are of length 0.
5	ndim	Number of axes / array dimensions.

Basic DataFrame Functionality

Sl. No.	Attribute	Description
6	shape	Returns a tuple representing the dimensionality of the DataFrame.
7	size	Number of elements in the NDFrame.
8	values	Numpy representation of NDFrame.
9	head()	Returns the first n rows.
10	tail()	Returns last n rows.

Descriptive Statistics

S.No.	Function	Description
1	count()	Number of non-null observations
2	sum()	Sum of values
3	mean()	Mean of Values
4	median()	Median of Values
5	mode()	Mode of values
6	std()	Standard Deviation of the Values
7	min()	Minimum Value
8	max()	Maximum Value
9	abs()	Absolute Value
10	prod()	Product of Values

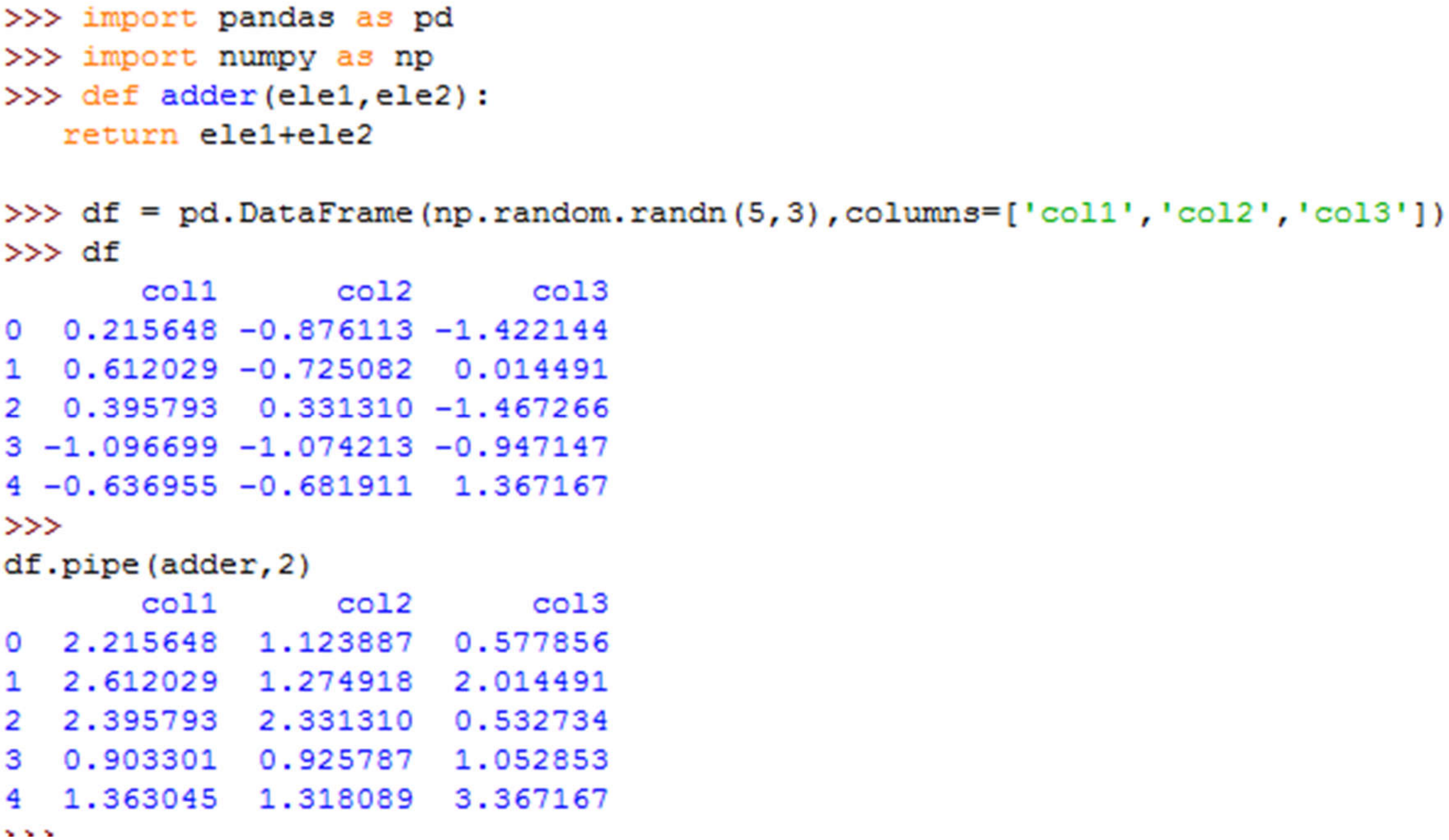
Descriptive Statistics

- Use of describe function
  - `df.describe(include=[‘number'])`
  - `df.describe(include=['object'])`
  - `df.describe(include=[‘all'])`

Function Application

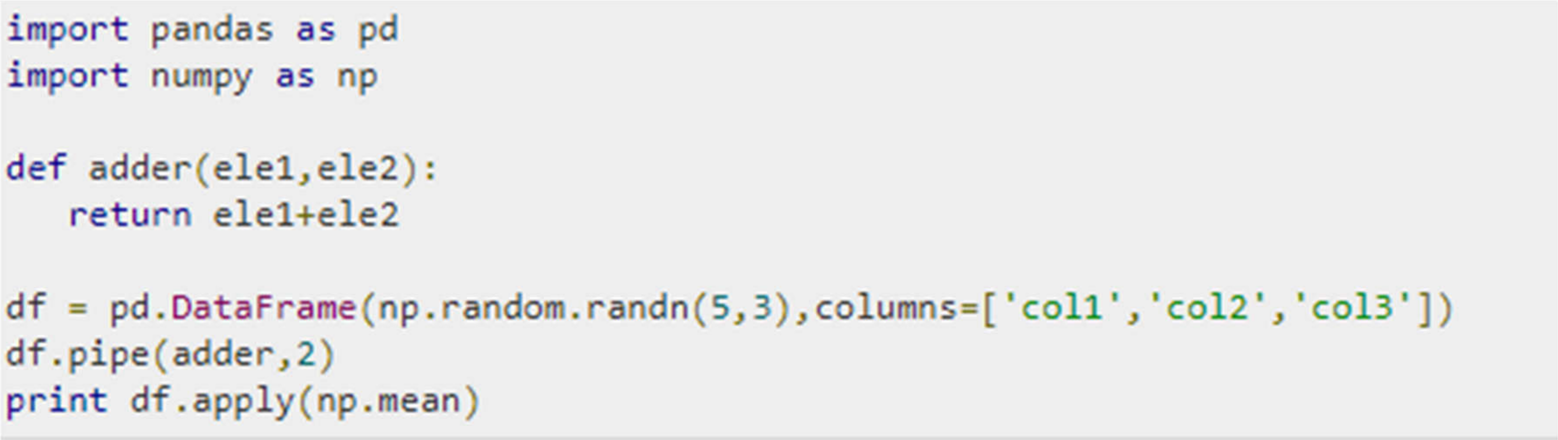
- Table wise Function Application
  - pipe()
- Row or Column Wise Function Application
  - apply()
- Element wise Function Application
  - applymap()

pipe



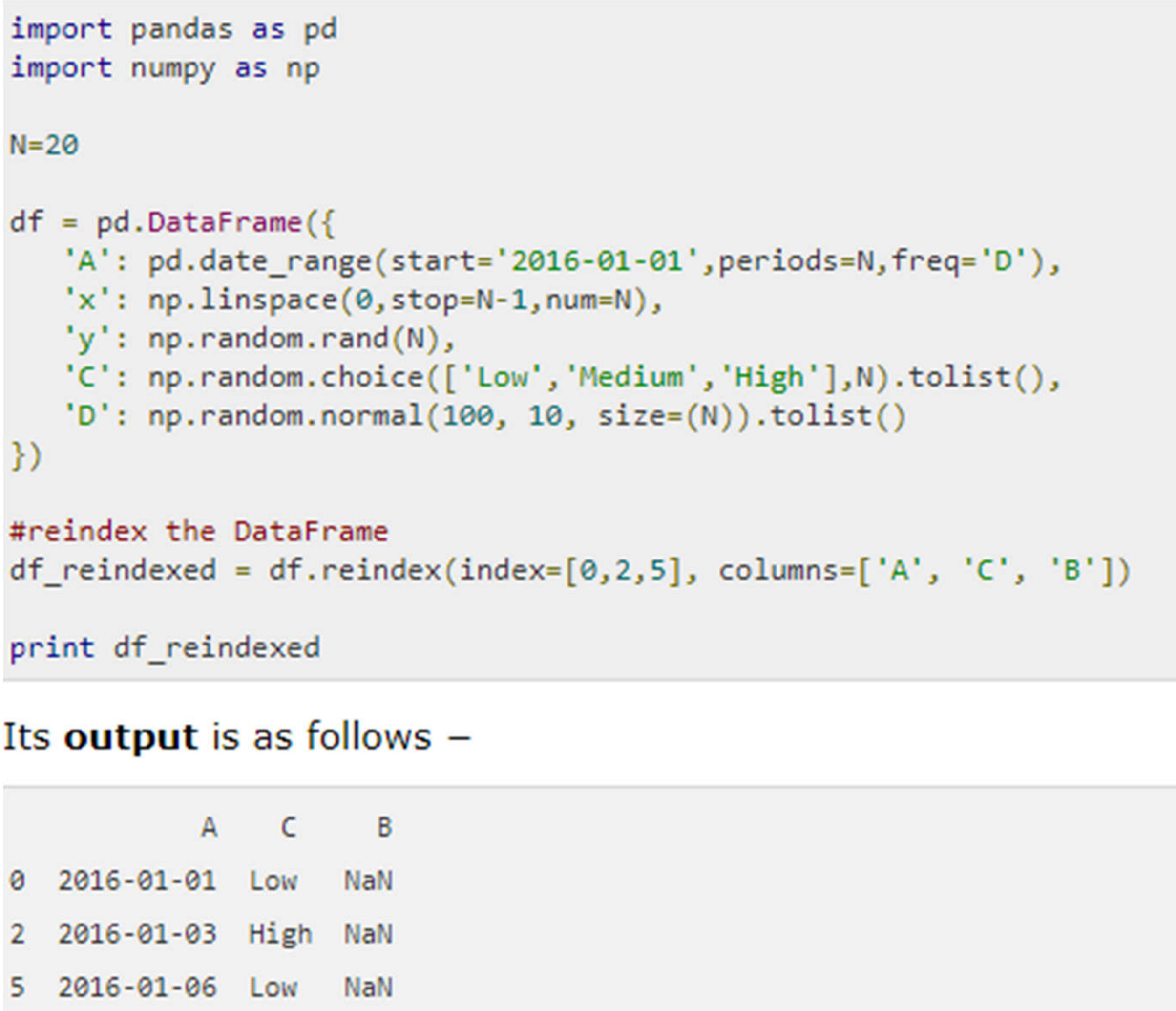
```
>>> import pandas as pd
>>> import numpy as np
>>> def adder(ele1,ele2):
	return ele1+ele2

>>> df = pd.DataFrame(np.random.randn(5,3),columns=['col1','col2','col3'])
>>> df
       col1      col2      col3
0  0.215648 -0.876113 -1.422144
1  0.612029 -0.725082  0.014491
2  0.395793  0.331310 -1.467266
3 -1.096699 -1.074213 -0.947147
4 -0.636955 -0.681911  1.367167
>>> 
df.pipe(adder,2)
       col1      col2      col3
0  2.215648  1.123887  0.577856
1  2.612029  1.274918  2.014491
2  2.395793  2.331310  0.532734
3  0.903301  0.925787  1.052853
4  1.363045  1.318089  3.367167
>>>
```

apply



```
import pandas as pd
import numpy as np

def adder(ele1,ele2):
   return ele1+ele2

df = pd.DataFrame(np.random.randn(5,3),columns=['col1','col2','col3'])
df.pipe(adder,2)
print df.apply(np.mean)
```

Reindexing



```
import pandas as pd
import numpy as np

N=20

df = pd.DataFrame({
   'A': pd.date_range(start='2016-01-01',periods=N,freq='D'),
   'x': np.linspace(0,stop=N-1,num=N),
   'y': np.random.rand(N),
   'C': np.random.choice(['Low','Medium','High'],N).tolist(),
   'D': np.random.normal(100, 10, size=(N)).tolist()
})

#reindex the DataFrame
df_reindexed = df.reindex(index=[0,2,5], columns=['A', 'C', 'B'])

print df_reindexed
```

Its **output** is as follows –

```
            A    C     B

0  2016-01-01  Low   NaN

2  2016-01-03  High  NaN

5  2016-01-06  Low   NaN
```

Iteration

- Iterating a DataFrame
- Iterating using
  - **iteritems()** − to iterate over the (key,value) pairs
  - **iterrows()** − iterate over the rows as (index,series) pairs
  - **itertuples()** − iterate over the rows as namedtuples

Sort

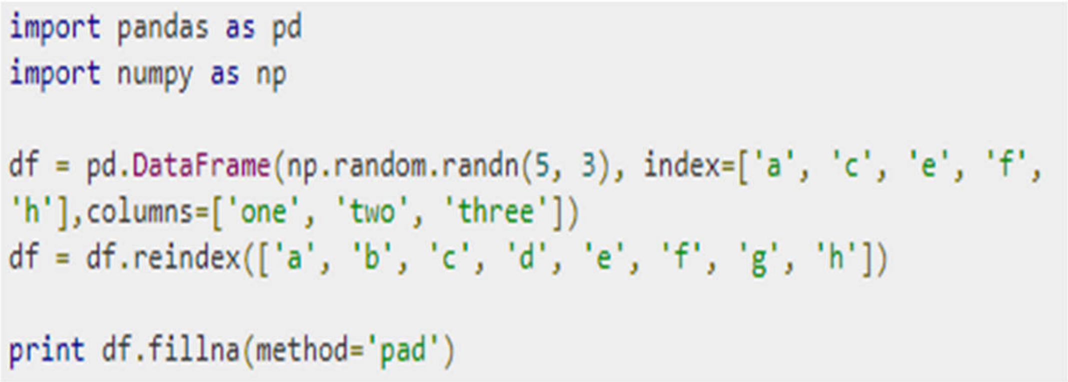
- sort_index()
- sort_index(ascending=False)
- sort_index(axis=1)
- sort_values(by=<columnName>)
- sort_values(by=[<columnName1>,<columnName2>])

Statistical Functions

- pct_change()
- cov()
- rank()

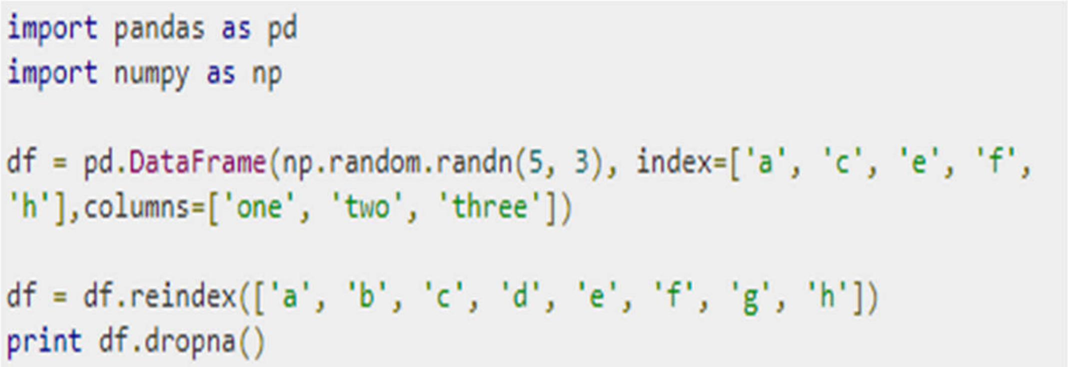
Handling Missing Values

- fillna()



```
import pandas as pd
import numpy as np

df = pd.DataFrame(np.random.randn(5, 3), index=['a', 'c', 'e', 'f',
'h'],columns=['one', 'two', 'three'])
df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])

print df.fillna(method='pad')
```

- dropna()

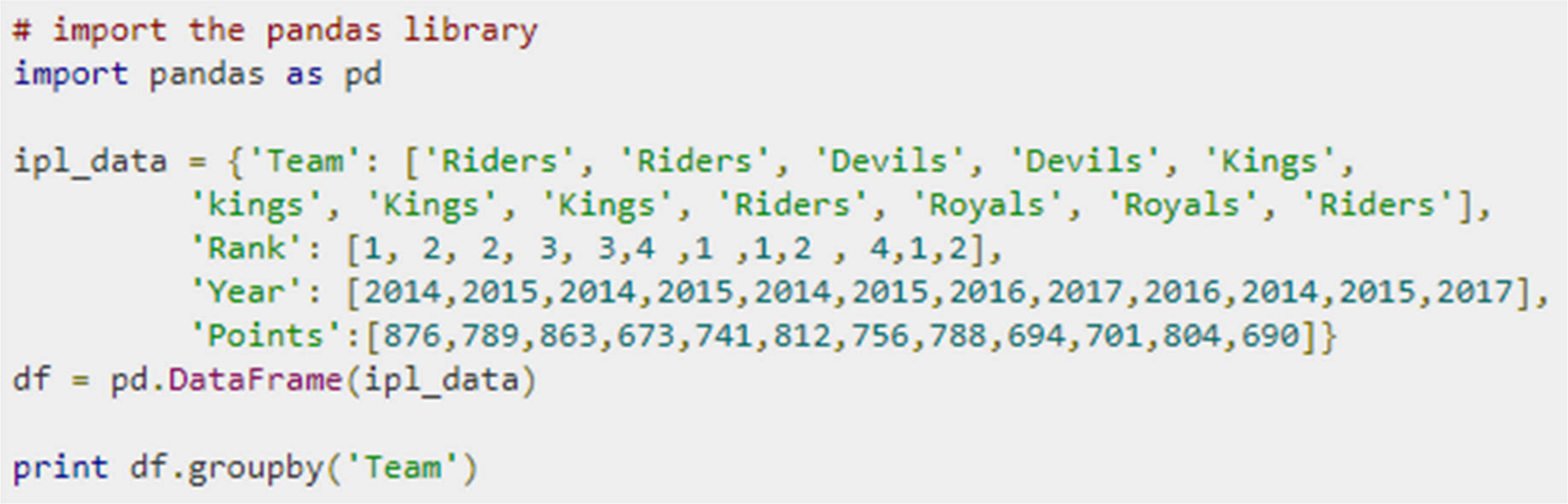


```
import pandas as pd
import numpy as np

df = pd.DataFrame(np.random.randn(5, 3), index=['a', 'c', 'e', 'f',
'h'],columns=['one', 'two', 'three'])

df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])
print df.dropna()
```

Group By

- Any **groupby** operation involves one of the following operations on the original object
  - **Splitting** the Object
  - **Applying** a function
  - **Combining** the results

Group By



```
# import the pandas library
import pandas as pd

ipl_data = {'Team': ['Riders', 'Riders', 'Devils', 'Devils', 'Kings',
         'kings', 'Kings', 'Kings', 'Riders', 'Royals', 'Royals', 'Riders'],
         'Rank': [1, 2, 2, 3, 3,4 ,1 ,1,2 , 4,1,2],
         'Year': [2014,2015,2014,2015,2014,2015,2016,2017,2016,2014,2015,2017],
         'Points':[876,789,863,673,741,812,756,788,694,701,804,690]}
df = pd.DataFrame(ipl_data)

print df.groupby('Team')
```

Group By

- Group by multiple columns
  - `df.groupby(['Team','Year']).groups`

- View groups
  - `df.groupby('Team').groups`

- Select group
  - `grouped.get_group(2014)`

Group By

- Aggregation
  - Use agg() function

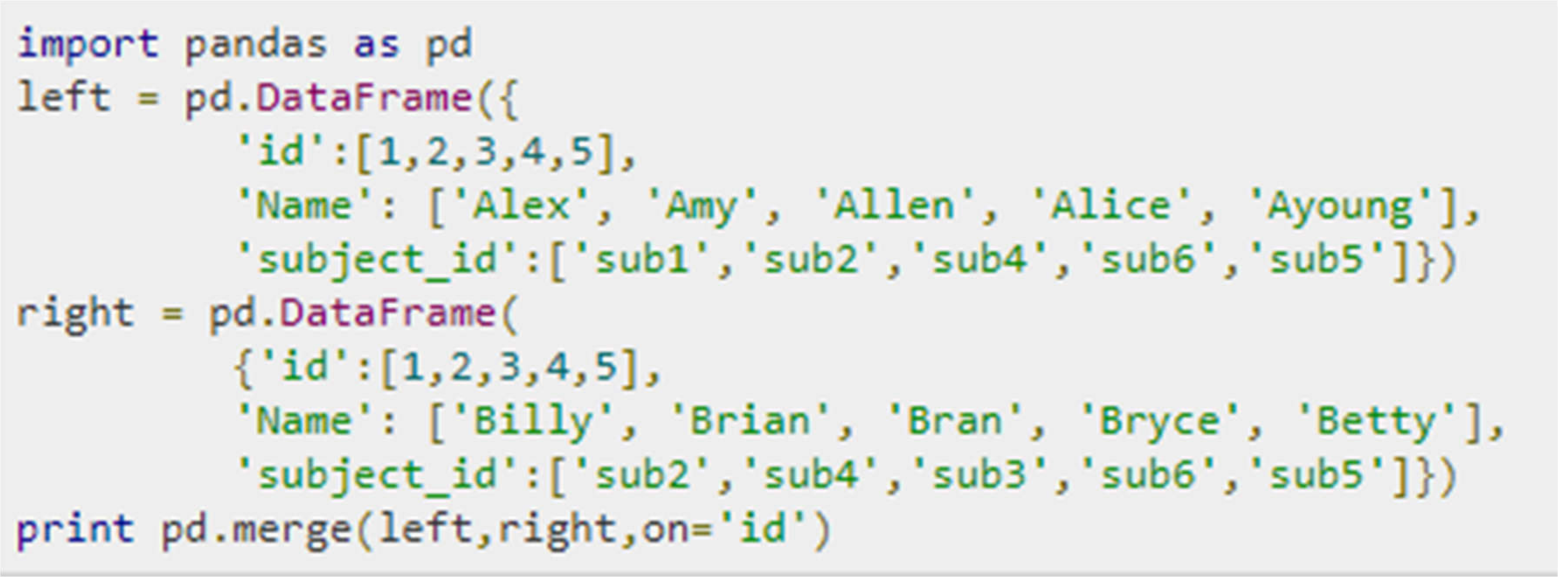
  -
    ```
    grouped['Points'].agg(np.mean)
    ```

  -
    ```
    grouped['Points'].agg([np.sum, np.mean, np.std])
    ```

- Filter

  -
    ```
    grouped.filter(lambda x: len(x) >= 3)
    ```

Merge



```
import pandas as pd
left = pd.DataFrame({
         'id':[1,2,3,4,5],
         'Name': ['Alex', 'Amy', 'Allen', 'Alice', 'Ayoung'],
         'subject_id':['sub1','sub2','sub4','sub6','sub5']})
right = pd.DataFrame(
         {'id':[1,2,3,4,5],
         'Name': ['Billy', 'Brian', 'Bran', 'Bryce', 'Betty'],
         'subject_id':['sub2','sub4','sub3','sub6','sub5']})
print pd.merge(left,right,on='id')
```

Merge

- Merge Two DataFrames on Multiple Keys

  -
    ```
    pd.merge(left,right,on=['id','subject_id'])
    ```

- Merge Using 'how' Argument

  -
    ```
    pd.merge(left, right, on='subject_id', how='left')
    ```

  -
    ```
    pd.merge(left, right, on='subject_id', how='right')
    ```

  -
    ```
    pd.merge(left, right, how='outer', on='subject_id')
    ```

  -
    ```
    pd.merge(left, right, on='subject_id', how='inner')
    ```

I/O

- `pd.read_csv("temp.csv")`
- `pd.read_csv("temp.csv",index_col=['S.No'])`
  - Using custom index
- `pd.read_csv("temp.csv", names=['a','b','c','d','e'],header=0)`

Reference

- https://www.tutorialspoint.com/python_pandas/
- https://pandas.pydata.org/
- http://pandas.pydata.org/pandas-docs/stable/tutorials.html

Thank You