# NUMPY , PANDAS AND MATPLOTLIB

# Advantages of Pandas

- Fast and efficient DataFrame object with default and customized indexing.
- Tools for loading data into in-memory data objects from different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of date sets.
- Label-based slicing, indexing and subsetting of large data sets.

# Advantages continued

- 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 types

- Pandas deals with the following three data structures:

1. Series
2. DataFrame
3. Panel

- These data structures are built on top of Numpy array, which means they are fast.

# Dimension and Description

- The best way to think of these data structures is that the higher dimensional data structure is a container of its lower dimensional data structure. For example, DataFrame is a container of Series, Panel is a container of DataFrame.

| Data Structure | Dimensions | Description |
|---|---|---|
| Series | 1 | 1D labeled homogeneous array, sizeimmutable. |
| Data Frames | 2 | General 2D labeled, size-mutable tabular structure with potentially heterogeneously typed columns. |
| Panel | 3 | General 3D labeled, size-mutable array. |

# NOTES

- All Pandas data structures are value mutable (can be changed) and except Series all are size mutable. Series is size immutable.
- DataFrame is widely used and one of the most important data structures. Panel is used much less.

# SERIES

- Series is a one-dimensional array like structure with homogeneous data. For example, the following series is a collection of integers 10, 23, 56, …

| 10 | 23 | 56 | 17 | 52 | 61 | 73 | 90 | 26 | 72 |
|---|---|---|---|---|---|---|---|---|---|

- Homogeneous data
- Size Immutable
- Values of Data Mutable

# DATAFRAME

- DataFrame is a two-dimensional array with heterogeneous data.

| Name | Age | Gender | Rating |
|---|---|---|---|
| Steve | 32 | Male | 3.45 |
| Lia | 28 | Female | 4.6 |
| Vin | 45 | Male | 3.9 |
| Katie | 38 | Female | 2.78 |

# Panel

- Panel is a three-dimensional data structure with heterogeneous data. It is hard to represent the panel in graphical representation. But a panel can be illustrated as a container of DataFrame.
- Heterogeneous data
- Size Mutable
- Data Mutable

# Creating a Series

- pandas.Series( data, index, dtype, copy)
- A series can be created using various inputs like:

1. Array
2. Dict
3. Scalar value or constant

# EXAMPLE

```
IMPORT PANDAS AS PD
S = PD.SERIES()
```

```
IMPORT PANDAS AS PD
IMPORT NUMPY AS NP
DATA = NP.ARRAY(['A','B','C','D'])
#S = PD.SERIES(DATA,INDEX=[100,101,102,103])
S = PD.SERIES(DATA)
PRINT (S)
```

# CREATING A SERIES FROM A DICT

```
DATA = {'A' : 0., 'B' : 1., 'C' : 2.}
S = PD.SERIES(DATA)
DATA = {'A' : 0., 'B' : 1., 'C' : 2.}
S = PD.SERIES(DATA,INDEX=['B','C','D','A'])
```

# CREATING A SERIES FROM A SCALAR

```
s = pd.Series(5, index=[0, 1, 2, 3])
```

Output:

```
0  5
1  5
2  5
3  5
dtype: int64
```

# ACCESSING DATA FROM SERIES WITH POSITION

```
S = PD.SERIES([1,2,3,4,5],INDEX = ['A','B','C','D','E'])
#RETRIEVE THE FIRST ELEMENT
PRINT (S[0])
S = PD.SERIES([1,2,3,4,5],INDEX = ['A','B','C','D','E'])
#RETRIEVE THE FIRST THREE ELEMENT
PRINT S[:3]
```

# RETRIEVE DATA USING LABEL (INDEX)

```
S = PD.SERIES([1,2,3,4,5],INDEX = ['A','B','C','D','E'])
#RETRIEVE A SINGLE ELEMENT
PRINT (S['A'])
S = PD.SERIES([1,2,3,4,5],INDEX = ['A','B','C','D','E'])
#RETRIEVE MULTIPLE ELEMENTS
PRINT S[['A','C','D']]
```

# Features of DataFrame

- Potentially columns are of different types
- Size – Mutable
- Labeled axes (rows and columns)
- Can Perform Arithmetic operations on rows and columns

# Creating a DataFrame

- pandas.DataFrame( data, index, columns, dtype, copy)
- A pandas DataFrame can be created using various inputs like:

1. Lists
2. dict
3. Series
4. Numpy ndarrays
5. Another DataFrame

# EXAMPLES

```
DF = PD.DATAFRAME()

DATA = [1,2,3,4,5]
DF = PD.DATAFRAME(DATA)

DATA = [['ALEX',10],['BOB',12],['CLARKE',13]]
DF = PD.DATAFRAME(DATA,COLUMNS=['NAME','AGE'])
```

# EXMAPLES CONTINUED

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

# CREATE A DATAFRAME FROM DICT OF NDARRAYS AND LISTS

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

```
data = {'Name':['Tom', 'Jack', 'Steve', 'Ricky'],'Age':[28,34,29,42]}
df = pd.DataFrame(data, index=['rank1','rank2','rank3','rank4'])
```

# CREATE A DATAFRAME FROM LIST OF DICTS

```
DATA = [{'A': 1, 'B': 2},{'A': 5, 'B': 10, 'C': 20}]
DF = PD.DATAFRAME(DATA)

DATA = [{'A': 1, 'B': 2},{'A': 5, 'B': 10, 'C': 20}]
DF = PD.DATAFRAME(DATA, INDEX=['FIRST', 'SECOND'])
```

# CONTINUED

```
data = [{'a': 1, 'b': 2},{'a': 5, 'b': 10, 'c': 20}]

#With two column indices, values same as dictionary keys
df1 = pd.DataFrame(data, index=['first', 'second'], columns=['a', 'b'])

#With two column indices with one index with other name
df2 = pd.DataFrame(data, index=['first', 'second'], columns=['a', 'b1'])
```

# CREATE A DATAFRAME FROM DICT OF SERIES

```
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)
```

# CONTINUED

```
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)
```

# COLUMN ADDITION

```
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)

# Adding a new column to an existing DataFrame object with column label by passing new series

print ("Adding a new column by passing as Series:")
df['three']=pd.Series([10,20,30],index=['a','b','c'])
print df

print ("Adding a new column using the existing columns in DataFrame:")
df['four']=df['one']+df['three']
```

# COLUMN DELETION

```
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']),
    'THREE' : PD.SERIES([10,20,30], INDEX=['A','B','C'])}

DF = PD.DATAFRAME(D)
PRINT ("OUR DATAFRAME IS:")
PRINT DF

# USING DEL FUNCTION
PRINT ("DELETING THE FIRST COLUMN USING DEL FUNCTION:")
DEL DF['ONE']
PRINT DF

# USING POP FUNCTION
PRINT ("DELETING ANOTHER COLUMN USING POP FUNCTION:")
DF.POP('TWO')
```

# Row Selection, Addition, and Deletion

- Rows can be selected by passing row label to a **loc** function.

```
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.loc['b'])
```

- The result is a series with labels as column names of the DataFrame. And, the Name of the series is the label with which it is retrieved.

# CONTINUED

- Rows can be selected by passing integer location to an **iloc** function.

```
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.iloc[2])
```

# SLICE ROWS

- Multiple rows can be selected using ‘ : ’ operator

```
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[2:4])
```

# ADDITION OF ROWS

```
DF = PD.DATAFRAME([[1, 2], [3, 4]], COLUMNS = ['A','B'])
DF2 = PD.DATAFRAME([[5, 6], [7, 8]], COLUMNS = ['A','B'])

DF = DF.APPEND(DF2)
```

# DELETION OF ROWS

```
import pandas as pd

df = pd.DataFrame([[1, 2], [3, 4]], columns = ['a','b'])
df2 = pd.DataFrame([[5, 6], [7, 8]], columns = ['a','b'])

df = df.append(df2)

# Drop rows with label 0
df = df.drop(0)
```

# SERIES FUNCTIONS

```
s = pd.Series(np.random.randn(4))
s = pd.Series(np.random.randn(4))
print ("The axes are:")
print (s.axes)
```

- Returns the list of the labels of the series

# CONTINUED

```
s = pd.Series(np.random.randn(4))
print ("Is the Object empty?")
print s.empty
```

- Returns a boolean if the series is empty or not.

# CONTINUED

- Ndim , returns dimension.
- Size , returns number of elements in the series.
- Values, returns actual elements in the series.
- Head() and tail(), returns values from both sides.

# DATAFRAME BASIC FUNCTIONALITIES

- T (TRANSPOSE OF A DATAFRAME)

```
d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Smith','Jack']),
    'Age':pd.Series([25,26,25,23,30,29,23]),
    'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8])}

# Create a DataFrame
df = pd.DataFrame(d)
print ("The transpose of the data series is:")
print (df.T)
```

# CONTINUED

- Axes
- Dtypes
- Empty
- Ndim
- Shape
- Size
- Values
- Head
- tails

# Descriptive Statistics

- Sum() , you can specify the axis
- Mean()
- Std()
- Describe()

# ROW OR COLUMN WISE FUNCTION APPLICATION

```
IMPORT PANDAS AS PD
IMPORT NUMPY AS NP

DF = PD.DATAFRAME(NP.RANDOM.RANDN(5,3),COLUMNS=['COL1','COL2','COL3'])
DF.APPLY(NP.MEAN)
PRINT DF.APPLY(NP.MEAN)
```

# ITERATING A DATAFRAME

- Iterating a DataFrame gives column names

```
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()
    })

FOR COL IN DF:
    PRINT (COL)
```

# CONTINUED

- To iterate over the rows of the DataFrame, we can use the following functions:

1. **iteritems()** – to iterate over the (key,value) pairs
2. **iterrows()** – iterate over the rows as (index,series) pairs
3. **itertuples()** – iterate over the rows as namedtuples

# ITERITEMS()

- Iterates over each column as key, value pair with label as key and column value as a Series object

```
import pandas as pd
import numpy as np

df = pd.DataFrame(np.random.randn(4,3),columns=['col1','col2','col3'])
for key,value in df.iteritems():
    print (key,value)
```

# ITERROWS()

- ITERROWS() RETURNS THE ITERATOR YIELDING EACH INDEX VALUE ALONG WITH A SERIES CONTAINING THE DATA IN EACH ROW.

```
IMPORT PANDAS AS PD

IMPORT NUMPY AS NP


DF = PD.DATAFRAME(NP.RANDOM.RANDN(4,3),COLUMNS = ['COL1','COL2','COL3'])

FOR ROW_INDEX,ROW IN DF.ITERROWS():

    PRINT (ROW_INDEX,ROW)
```

# ITERTUPLES()

- ITERTUPLES() METHOD WILL RETURN AN ITERATOR YIELDING A NAMED TUPLE FOR EACH ROW IN THE DATAFRAME. THE FIRST ELEMENT OF THE TUPLE WILL BE THE ROW'S CORRESPONDING INDEX VALUE, WHILE THE REMAINING VALUES ARE THE ROW VALUES.

```
IMPORT PANDAS AS PD

IMPORT NUMPY AS NP


DF = PD.DATAFRAME(NP.RANDOM.RANDN(4,3),COLUMNS = ['COL1','COL2','COL3'])

FOR ROW IN DF.ITERTUPLES():

    PRINT (ROW)
```