Name :- Pranav Trivedi
Branch :- TEIT-2
Roll No :- 61
Batch :- I6

Experiment No : 1

AIM : Introduction to Pandas , Environment setup ,Introduction to Data Structure, introduction to Series and Data Frame

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. The name Pandas is derived from the word Panel Data – an Econometrics from Multidimensional data.

In 2008, developer Wes McKinney started developing pandas when in need of high performance, flexible tool for analysis of data.

Prior to Pandas, Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data — load, prepare, manipulate, model, and analyze.

Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Key Features 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.**
- **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.**

Python Pandas - Environment Setup

Standard Python distribution doesn't come bundled with Pandas module. A lightweight alternative is to install NumPy using popular Python package installer, pip.

```
pip install pandas
```

If you install Anaconda Python package, Pandas will be installed by default with the following −

Windows

- **Anaconda (from https://www.continuum.io) is a free Python distribution for SciPy stack. It is also available for Linux and Mac.**
- **Canopy (https://www.enthought.com/products/canopy/) is available as free as well as commercial distribution with full SciPy stack for Windows, Linux and Mac.**
- **Python (x,y) is a free Python distribution with SciPy stack and Spyder IDE for Windows OS. (Downloadable from http://python-xy.github.io/)**

Linux

Package managers of respective Linux distributions are used to install one or more packages in SciPy stack.

For Ubuntu Users

```
sudo apt-get install python-numpy python-scipy python-matplotlibipythonipythonnotebook
python-pandas python-sympy python-nose
```

For Fedora Users

```
sudo yum install numpyscipy python-matplotlibipython python-pandas sympy
python-nose atlas-devel
```

Introduction to Data Structures

Pandas deals with the following three data structures −

- **Series**
- **DataFrame**
- **Panel**

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

Dimension & 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.**

Building and handling two or more dimensional arrays is a tedious task, burden is placed on the user to consider the orientation of the data set when writing functions. But using Pandas data structures, the mental effort of the user is reduced.

For example, with tabular data (DataFrame) it is more semantically helpful to think of the index (the rows) and the columns rather than axis 0 and axis 1.

Mutability

All Pandas data structures are value mutable (can be changed) and except Series all are size mutable. Series is size immutable.

Note − 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**

Key Points

- **Homogeneous data**
- **Size Immutable**
- **Values of Data Mutable**

DataFrame

DataFrame is a two-dimensional array with heterogeneous data. For example,

Name	Age	Gender	Rating
Steve	**32**	**Male**	**3.45**
Lia	**28**	**Female**	**4.6**

Vin	**45**	**Male**	**3.9**
Katie	**38**	**Female**	**2.78**

The table represents the data of a sales team of an organization with their overall performance rating. The data is represented in rows and columns. Each column represents an attribute and each row represents a person.

Data Type of Columns

The data types of the four columns are as follows −

Column	Type
Name	**String**
Age	**Integer**
Gender	**String**
Rating	**Float**

Key Points

- **Heterogeneous data**
- **Size Mutable**
- **Data Mutable**

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.

Key Points

- **Heterogeneous data**
- **Size Mutable**
- **Data Mutable**

Python Pandas - Series

eries is a one-dimensional labeled array capable of holding data of any type (integer, string, float, python objects, etc.). The axis labels are collectively called index.

pandas.Series

A pandas Series can be created using the following constructor −

```
pandas.Series( data, index, dtype, copy)
```

The parameters of the constructor are as follows −

Sr.No	Parameter & Description
1	**Data** **data takes various forms like ndarray, list, constants**
2	**Index** **Index values must be unique and hashable, same length as data. Default np.arrange(n) if no index is passed.**
3	**Dtype** **dtype is for data type. If None, data type will be inferred**
4	**Copy** **Copy data. Default False**

A series can be created using various inputs like −

- **Array**
- **Dict**
- **Scalar value or constant**

Output :-

```
#create a series from dict
data = {'a':1,'b':2,'c':3}
s = pd.Series(data)
print(s)
```

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

```
[9] #Create a Series from Scalar
s = pd.Series(5, index=[0, 1, 2, 3])
print(s)
```

```
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'])
print(s[1:4])
```

```
b    2
c    3
d    4
dtype: int64
```

```
[11] #Retrieve the last three elements.
s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])
print(s[-3:])
```

```
c    3
d    4
e    5
dtype: int64
```

```
[15] #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'])
```

```
1
```

```
#Retrieve Data Using Label (Index)
s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])
print(s[['a','b','e']])
```

```
a    1
b    2
e    5
dtype: int64
```

```
[18] #create an empty dataframe
df = pd.DataFrame()
print(df)
```

```
Empty DataFrame
Columns: []
Index: []
```

```
[19] #Create a DataFrame from Lists
data  = [1,2,3,4,5]
df = pd.DataFrame(data)
print(df)
```

```
   0
0  1
1  2
2  3
3  4
4  5
```

```
data = [['Alex',12],['Bob',13],['Drew',14]]
df = pd.DataFrame(data,columns=['Name','Age'])
print(df)
```

```
   Name  Age
0  Alex   12
1   Bob   13
2  Drew   14
```

```
[22] #Create a DataFrame from Dict of ndarrays / Lists
data = {'Name':['Drew','Alex','Matt'],'Age':[32,36,26]}
df = pd.DataFrame(data)
print(df)
```

```
   Name  Age
0  Drew   32
1  Alex   36
2  Matt   26
```

```
[25]  #Create a DataFrame from List of Dicts
data = [{'a': 1, 'b': 2},{'a': 5, 'b': 10, 'c': 20}]
df = pd.DataFrame(data)
print(df)
```

```
   a   b     c
0  1   2   NaN
1  5  10  20.0
```

```
#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)
print(df)
```

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

```
[29] 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)
```

```
a    1.0
b    2.0
c    3.0
d    NaN
Name: one, dtype: float64
```

```
#adding column
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("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']

print(df)
```

```
Adding a new column by passing as Series:
   one  two  three
a  1.0    1   10.0
b  2.0    2   20.0
c  3.0    3   30.0
d  NaN    4    NaN
Adding a new column using the existing columns in DataFrame:
   one  two  three  four
a  1.0    1   10.0  11.0
b  2.0    2   20.0  22.0
c  3.0    3   30.0  33.0
d  NaN    4    NaN   NaN
```

+ Code + Text

```
[31] 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)

    print('Delete column one')

    del df['one']
    print(df)
```

```
a    1.0
b    2.0
c    3.0
d    NaN
Name: one, dtype: float64
Delete column one
   two
a    1
b    2
c    3
d    4
```

```
#Selecting rows
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'])
```

```
one    2.0
two    2.0
Name: b, dtype: float64
```

```
[45] 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[0:2,0])
```

```
a    1.0
b    2.0
Name: one, dtype: float64
```

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

df = df.append(df2)
print(df)
```

```
   a  b
0  1  2
1  3  4
0  5  6
1  7  8
```

```
[49] #deletion of rows
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)
df = df.drop(columns=['one'],axis=1)
print(df)
```

```
   two
a    1
b    2
c    3
d    4
```