Pandas

Pandas deals with the following three data structures.
1)series
2)Data frames
3)Panel These three data structures are faster than numpy array.

Series: It is a one-dimensional array like structure with homogeneous data.For example the following series is a collection of integers

10 23 56 17 52 61 73 90 14

Dataframe: Dataframe is a two dimensional array with heterogeneous data.For ex,

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 . A panel is a container of dataframe.

Series: A Series can be created using various inputs like
1)Array
2)Dict
3)Scalar value or constant

Create an empty Series: A basic series which can be created is an empty series.

```
import pandas as pd
s=pd.Series()
print(s)
```

Series([], dtype: float64)

Creating a series from ndarray:
If data is an ndarray then index passed must be of the same length.If no index is passed ,then by default index will be range(n)

```
import pandas as pd
import numpy as np
data=np.array(['a','b','c','d'])  # no index is passed.So index ranges from 0 to 3
s=pd.Series(data)
print(s)
```

```
0  a
1  b
2  c
3  d
dtype: object
```

```
import pandas as pd
import numpy as np
data=np.array(['a','b','c','d'])
s=pd.Series(data,index=[100,101,102,103])  # index is passed.
print(s)
```

```
100  a
101  b
102  c
103  d
dtype: object
```

Create a Series from Dict: A dict can be passed as input and if no index is specified,then the dict keys are taken in a sorted order to construct index.If index is passed the values in data corresponding to the labels in the index will be pulled out.

```
data={'a':0,'b':1,'c':2}
s=pd.Series(data)
print(s)
```

```
a  0
b  1
c  2
dtype: int64
```

```
data={'a':0,'b':1,'c':2}
```

```
s=pd.Series(data,index=['b','c','d','a'])
print(s)
```

```
b   1.0
c   2.0
d   NaN
a   0.0
dtype: float64
```

Creating a Series from scalar: If data is a scalar value, an index must be provided. The value will be repeated to match the length of index.

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

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

Accessing data from series with position: counting strats from zero for the array, which means that the first element is stored at zeroth position and so on.

```
s=pd.Series([1,2,3,4],index=['a','b','c','d'])
print(s['a'])
print(s[:3])
print(s[-3:])
```

```
1
a   1
b   2
c   3
dtype: int64
b   2
c   3
d   4
dtype: int64
```

Retrieve data using label(index):

```
s=pd.Series([1,2,3,4],index=['a','b','c','d'])
print(s['a']) # retrive a single element using index label value
```

```
print(s[['a','b','d']]) #retrive multiple elements using index label value
```

```
1
a  1
b  2
d  4
dtype: int64
```

```
import numpy as np
s1=pd.Series(np.random.rand(4),index=['a','b','c','d'])
print(s1)
```

```
a  0.407940
b  0.447602
c  0.434150
d  0.514236
dtype: float64
```

```
import numpy as np
s1=pd.Series(np.random.rand(4))
print(s1)
```

```
0  0.399884
1  0.940370
2  0.294637
3  0.714556
dtype: float64
```

```
#Missing data: The library isnull() is used to detect missing data.
s=pd.Series({'001':'Nam','002':'mary','003':'peter'},
        index=['002','001','024','065'])
print(s)
#pd.isnull(s)
```

```
002   mary
001    Nam
024    NaN
065    NaN
dtype: object
```

Dataframe: A dataframe is a two dimensionals data structure. A pandas Dataframecan be created using the following constructor.

pandas.DataFrame(data,index,columns,dtype,copy)

here, data-data taskes various forms like,ndaray,series,map,lists,dict,constants and also another dataframe.
index-for the row labels,the index to be used for the resulting frame is optional default. np.arrange(n) if no index is passed.
columns- for columns labels the optional default syntax is np.arange(n) if no index is passed.
dtype- datatype of each column
copy-This command is used for copying of data,default is false.

Create Dataframe: A pandas dataframe canbe created using various inputs like
1)lists 2) Dict 3) Series 4) Numpy ndarrays. 5) Another dataframe

Create an empty DataFrame: A basic Dataframe,which can be created is an empty Dataframe.

```
import pandas as pd
df=pd.DataFrame()
print(df)
```

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

Creating dataframe from lists:

The dataframe canbe created using a single list or a list of lists

```
import pandas  as pd
data=[1,2,3,4,5]
df=pd.DataFrame(data,index=[4,5,6,7,8])
print(df)
```

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

```
import pandas  as pd
data=[1,2,3,4,5]
df=pd.DataFrame(data,index=['a','b','c','d','e'],columns=['x'])
print(df)
```

```
   x
a  1
b  2
c  3
d  4
e  5
```

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

```
     Name  Age
0    Alex   10
1     Bob   12
2  Clarke   13
```

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

```
     Name   Age
0    Alex  10.0
1     Bob  12.0
2  Clarke  13.0
```

```
import pandas as pd
data = [['Alex',10],['Bob',12],['Clarke',13]]
df = pd.DataFrame(data,index=['s1','s2','s3'],columns=['Name','Age'])
```

```
print(df)
```

```
    Name  Age
s1   Alex   10
s2    Bob   12
s3  Clarke   13
```

Creating a DataFrame from Dict of ndarray/Lists:

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

```
    Name  Age
0    Tom   28
1   Jack   34
2  Steve   29
3  Ricky   42
```

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

```
       Name  Age
rank1   Tom   28
rank2  Jack   34
rank3 Steve   29
rank4 Ricky   42
```

Create a DataFrame from list of Dicts: List of Dictionries can be passed as input data to create a DataFrame. The Dictionary keys are by default taken as column names

```
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
```

```
data=[{'a':1,'b':2},{'a':5,'b':10,'c':20}]
df=pd.DataFrame(data,index=['first','second'])
print(df)
print(df.index)  #gives index labels
print(df.c)  #or df['c'] gives particular label
```

```
        a   b    c
first   1   2   NaN
second  5   10  20.0
Index(['first', 'second'], dtype='object')
first     NaN
second    20.0
Name: c, dtype: float64
```

```
data=[{'a':1,'b':2},{'a':5,'b':10,'c':20}]
df1=pd.DataFrame(data,index=['first','second'],columns=['a','b'])
df2=pd.DataFrame(data,index=['first','second'],columns=['a','b1'])
print(df1)
print(".....")
print(df2)
```

```
        a   b
first   1   2
second  5   10
.....
        a  b1
first   1 NaN
second  5 NaN
```

Create a data frame 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
```

```
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
```

```
#column addition
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...")
df['three']=pd.Series([10,20,30],index=['a','b','c'])
print(df)
df['four']=df['one']+df['three']
print(df)
```

```
Adding a new column...
  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
  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
```

```
#column deletion
del df['one']
df.pop('two')
print(df)
```

```
   three  four
a   10.0  11.0
b   20.0  22.0
c   30.0  33.0
d    NaN   NaN
```

Row selection, addition, deletion:

```
#selection by label
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)
print(df.loc['b'])  #loc gets rows  with particular labels from the index.
print(df.loc['d'])
```

```
   one  two
a  1.0    1
b  2.0    2
c  3.0    3
d  NaN    4
one    2.0
two    2.0
Name: b, dtype: float64
one    NaN
two    4.0
Name: d, dtype: float64
```

In [45]:

```
print(df.iloc[2])  # iloc gets rows  at particular positions in the index (so it only takes integers)
```

```
one    3.0
two    3.0
Name: c, dtype: float64
```

In [49]:

```
#Addition of rows:
```

```
df1=pd.DataFrame([[1,2],[3,4]],columns=['a','b'])
df2=pd.DataFrame([[5,6],[7,8]],columns=['a','b'])
df1=df1.append(df2)
print(df1)
print(df1['a'])
print(df1.iloc[2])
#print(df1.index)   #gives index labels
```

```
   a  b
0  1  2
1  3  4
0  5  6
1  7  8
0    1
1    3
0    5
1    7
Name: a, dtype: int64
a    5
b    6
Name: 0, dtype: int64
```

Deletion of rows:

Use index label to delete or drop rows from a dataframe. If label is duplicated then multiple rows will be deleted.

```
df1=df1.drop(0)
print(df1)
```

```
   a  b
1  3  4
1  7  8
```

```
#creating a dataframe from csv files
df4=pd.read_csv('Person.csv')
print(df4)
```

```
    Name  Age   Career Province Gender
0  Peter   19   Pupil       TN      M
1   Mary   16  student       Sg      F
2    Mai   31   Nurse       SG      F
```

Reindexing and altering labels: changes the row labels and column labels of a DataFrame. To reindex means to conform the data to match a given set of labels along a particular axis.When reindexed labels do not exist in the data object, a default value of NaN will be automatically assigned to the position.

```
# for 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])  # index is passed.
print(s)
print(s.reindex([0,2,'x',3]))
print(s)
```

```
100   a
101   b
102   c
103   d
dtype: object
0   NaN
2   NaN
x   NaN
3   NaN
dtype: object
100   a
101   b
102   c
103   d
dtype: object
```

```
df4.reindex(index=[103,102,101,100])
```

	Name	Age	Career	Province	Gender
103	NaN	NaN	NaN	NaN	NaN
102	NaN	NaN	NaN	NaN	NaN
101	NaN	NaN	NaN	NaN	NaN
100	NaN	NaN	NaN	NaN	NaN

umn labels of a DataFrame.
long a particular axis. When
ill be automatically

```
df4.reindex(index=[0,2,'b',3].
        columns=['Name','Age','Career','p','s'])
```

	Name	Age	Career	p	s
0	Peter	19.0	Pupil	NaN	NaN
2	Mai	31.0	Nurse	NaN	NaN
b	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN

Head and tail: for large data objects ,we can not view or load all informationof the objects.Pandas support functions that allow us to isnspect a small sample.

```
s7=pd.Series(np.random.rand(1000))
#s7.head(4) #displays frst 5 lines
s7.tail() #dispays last 5 lines
```

```
995    0.824779
996    0.674442
997    0.901425
998    0.515911
999    0.226065
dtype: float64
```

Binary operations:

```
df5=pd.DataFrame(np.arange(9).reshape(3,3),columns=['a','b','c'])
print(df5)
```

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

```
df6=pd.DataFrame(np.arange(8).reshape(2,4),columns=['a','b','c','d'])
print(df6)
```

```
   a  b  c  d
0  0  1  2  3
1  4  5  6  7
```

```
#addition
```

```
#print(df5+df6)
df7=df5+df6
df7
```

	a	b	c	d
0	0.0	2.0	4.0	NaN
1	7.0	9.0	11.0	NaN
2	NaN	NaN	NaN	NaN

```
df7=df5.add(df6,fill_value=0)
print(df7)
```

```
     a    b     c    d
0  0.0  2.0   4.0  3.0
1  7.0  9.0  11.0  7.0
2  6.0  7.0   8.0  NaN
```

```
#comparision operators
df5.eq(df6)
```

	a	b	c	d
0	True	True	True	False
1	False	False	False	False
2	False	False	False	False

```
# Functional Statistics
#sum()
print(df5)
print(df5.sum())
print(df7.sum(skipna=True))
```

```
   a  b  c
0  0  1  2
1  3  4  5
2  6  7  8
a    9
b   12
```

```
c   15
dtype: int64
a    7.0
b   11.0
c   15.0
d    0.0
dtype: float64
```

```
df5.describe() #gives most of the statistical information
```

	a	b	c
count	3.0	3.0	3.0
mean	3.0	4.0	5.0
std	3.0	3.0	3.0
min	0.0	1.0	2.0
25%	1.5	2.5	3.5
50%	3.0	4.0	5.0
75%	4.5	5.5	6.5
max	6.0	7.0	8.0

```
df5.describe(percentiles=[0.5,0.8])
```

	a	b	c
count	3.0	3.0	3.0
mean	3.0	4.0	5.0
std	3.0	3.0	3.0
min	0.0	1.0	2.0
50%	3.0	4.0	5.0
80%	4.8	5.8	6.8
max	6.0	7.0	8.0

```
#Sorting:
df7=pd.DataFrame(np.arange(12).reshape(3,4),columns=['b','d','a','c'],index=['x','y','z'])
print(df7)
df7.sort_index(axis=1)
```

```
   b  d  a  c
x  0  1  2  3
```

y 4 5 6 7
z 8 9 10 11

	a	b	c	d
x	2	0	3	1
y	6	4	7	5
z	10	8	11	9

Computational Tools:

Working with missing data:

```
s=pd.Series({'001':'Nam','002':'mary','003':'peter'},
        index=['002','001','024','065'])
print(s)
s.dropna()
```

```
002    mary
001     Nam
024     NaN
065     NaN
dtype: object

002    mary
001     Nam
dtype: object
```

```
s1=pd.Series(np.random.randint(4,size=4))
s2=pd.Series(np.random.randint(4,size=4))
print(s1)
print(s2)
```

```
0   2
1   0
2   0
3   0
dtype: int32
0   3
1   3
2   3
3   2
dtype: int32
```

Panel: A panel is a 3D container of data. The names for the 3 axes are intended to give some semantic meaning to describing operations involving panel data. They are
items: axis0, each item corresponds to a dataframe cotained inside.
major_axis: axis1, it is the index(rows) of each of the Dataframes.
minor-axis:axis2, it is the columns of each of the Dataframes.
Panel creation: A Panel can be created using the following constructor. Syntax:
pandas.Panel(data,items,major_axis,minor_axis,dtype,copy)

```
#creating  an empty panel
import pandas as pd
import numpy as np
data=np.random.rand(2,4,5)
p=pd.Panel(data)
print(p)
#print(data)
```

```
<class 'pandas.core.panel.Panel'>
Dimensions: 2 (items) x 4 (major_axis) x 5 (minor_axis)
Items axis: 0 to 1
Major_axis axis: 0 to 3
Minor_axis axis: 0 to 4
```

```
#Panel creation from dictof DataFrame objects
data={'Items1':pd.DataFrame(np.random.randn(4,3)),
    'Items2':pd.DataFrame(np.random.randn(4,2))}
p=pd.Panel(data)
print(p)
print(p['Items2'])
```

```
<class 'pandas.core.panel.Panel'>
Dimensions: 2 (items) x 4 (major_axis) x 3 (minor_axis)
Items axis: Items1 to Items2
Major_axis axis: 0 to 3
Minor_axis axis: 0 to 2
        0         1         2
0  0.942150  0.512720 NaN
1 -0.346888 -1.382047 NaN
2  1.211330  0.857969 NaN
```

3 1.043809 0.114124 NaN

```
df1=pd.DataFrame(np.arange(12).reshape(4,3),columns=['a','b','c'])
df2=pd.DataFrame(np.arange(9).reshape(3,3),columns=['a','b','c'])
panel2=pd.Panel({'item1':df1,'item2':df2})
print(panel2)
```

```
<class 'pandas.core.panel.Panel'>
Dimensions: 2 (items) x 4 (major_axis) x 3 (minor_axis)
Items axis: item1 to item2
Major_axis axis: 0 to 3
Minor_axis axis: a to c
```

```
#Each item in a Panel is a DataFrame.We can select an item, by item name
panel2['item1']
```

	a	b	c
0	0	1	2
1	3	4	5
2	6	7	8
3	9	10	11

If we want to select data via an axis or data posision,we can use the ix method like on Series or DataFrame

```
panel2.ix[:,1:3,['b','c']]
```

```
<class 'pandas.core.panel.Panel'>
Dimensions: 2 (items) x 3 (major_axis) x 2 (minor_axis)
Items axis: item1 to item2
Major_axis axis: 1 to 3
Minor_axis axis: b to c
```

```
print(panel2.ix[: ,2 ,:])
```

```
   item1  item2
a      6    6.0
b      7    7.0
c      8    8.0
```