teleuniv

Innovative Interactive Immersive

Python: Pandas

Session - 5

Previous sessions:

Data Types, Collections

Control Statements, Operators

This Session:

Pandas

Introduction

Pandas : Panel Data System

Powerful and productive Python data analysis and management library

Rich data structures and functions to make working with structured data fast, easy, and expressive

Flexible data manipulation capabilities of **spreadsheets and relational databases**

Sophisticated indexing functionality

slice, dice, perform aggregations, select subsets of data

The ideal tool for Data Scientist

Munging data

Cleaning data

Analyzing data

Modeling data

Organizing the results of the analysis into a form suitable for plotting or tabular display

Types Of Pandas Data Structure

Pandas deals with the following three data structures –

1. Series
2. DataFrame
3. Panel

Dimension & Description:

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.

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, …

Data	10	14	15	25	30	45	55
Index	0	1	2	3	4	5	6

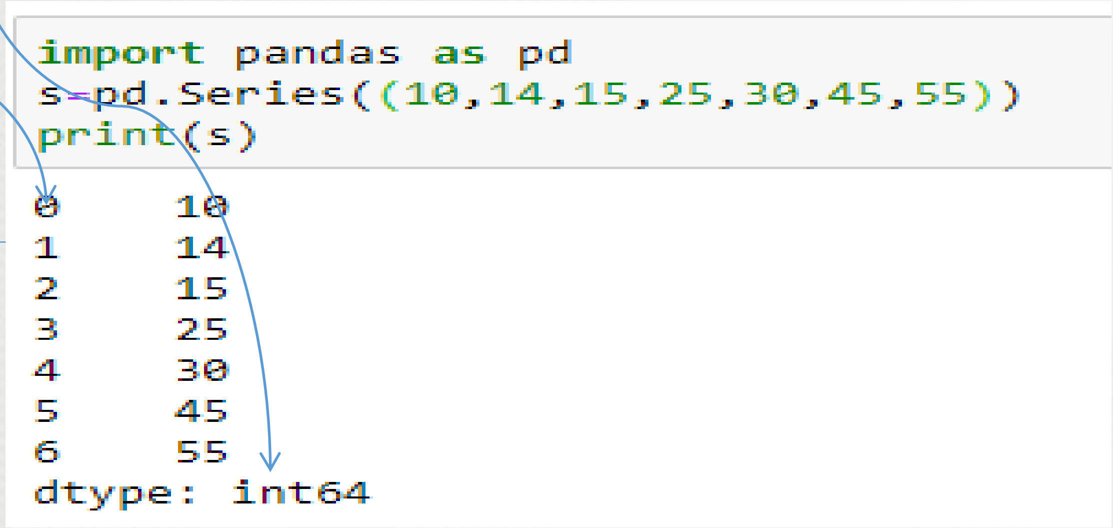
pandas. Series(data, index, dtype)

Key Points :

Homogeneous data

Size Immutable

Values of Data Mutable



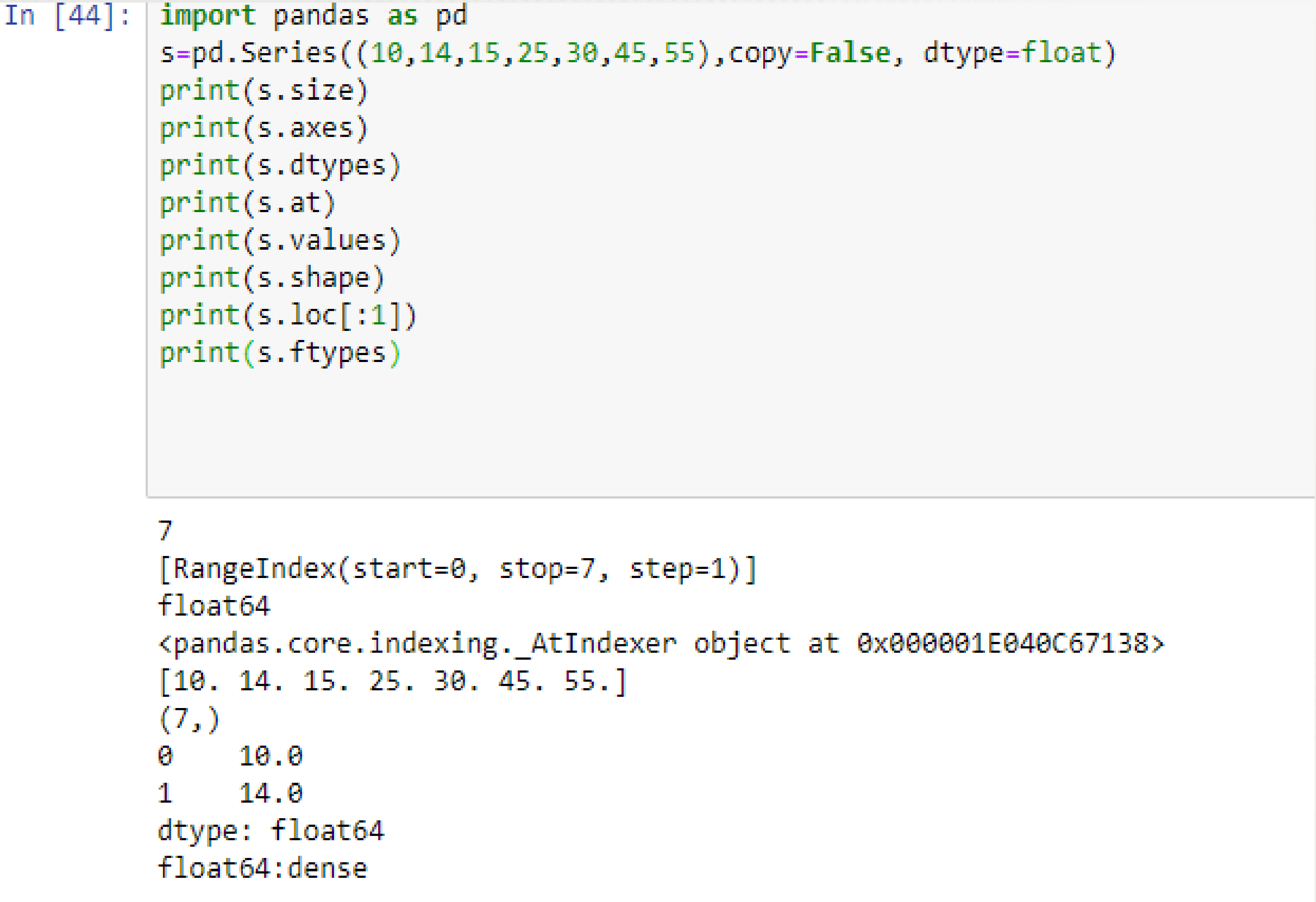
```
import pandas as pd
s=pd.Series((10,14,15,25,30,45,55))
print(s)
```

```
0      10
1      14
2      15
3      25
4      30
5      45
6      55
dtype: int64
```

Series

Attributes:

- size
- axes
- dtypes
- at
- values
- shape
- ftypes
- Loc
- etc…

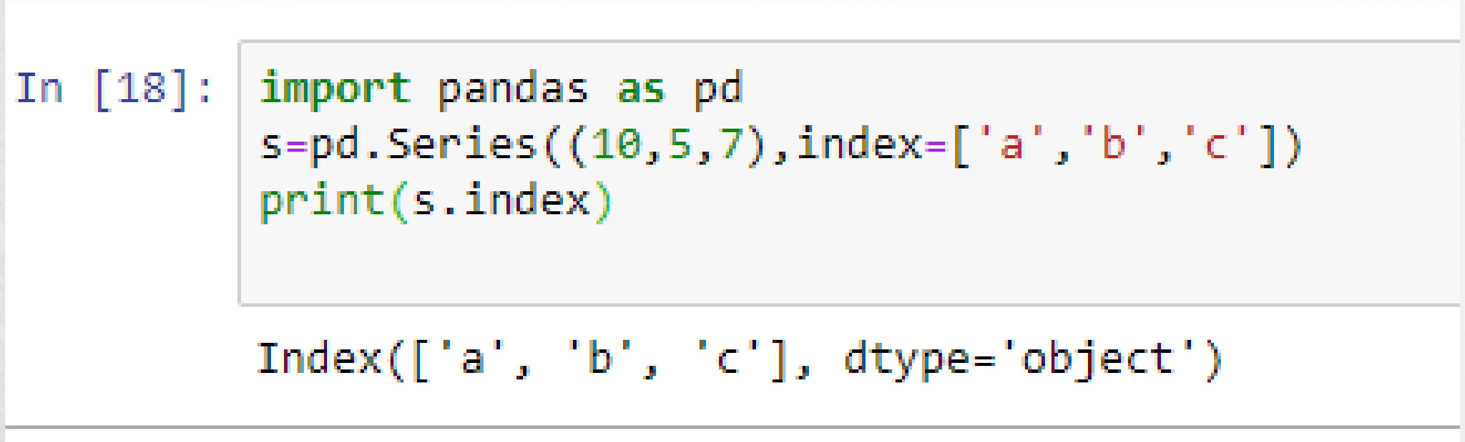


```
In [44]: import pandas as pd
         s=pd.Series((10,14,15,25,30,45,55),copy=False, dtype=float)
         print(s.size)
         print(s.axes)
         print(s.dtypes)
         print(s.at)
         print(s.values)
         print(s.shape)
         print(s.loc[:1])
         print(s.ftypes)
```

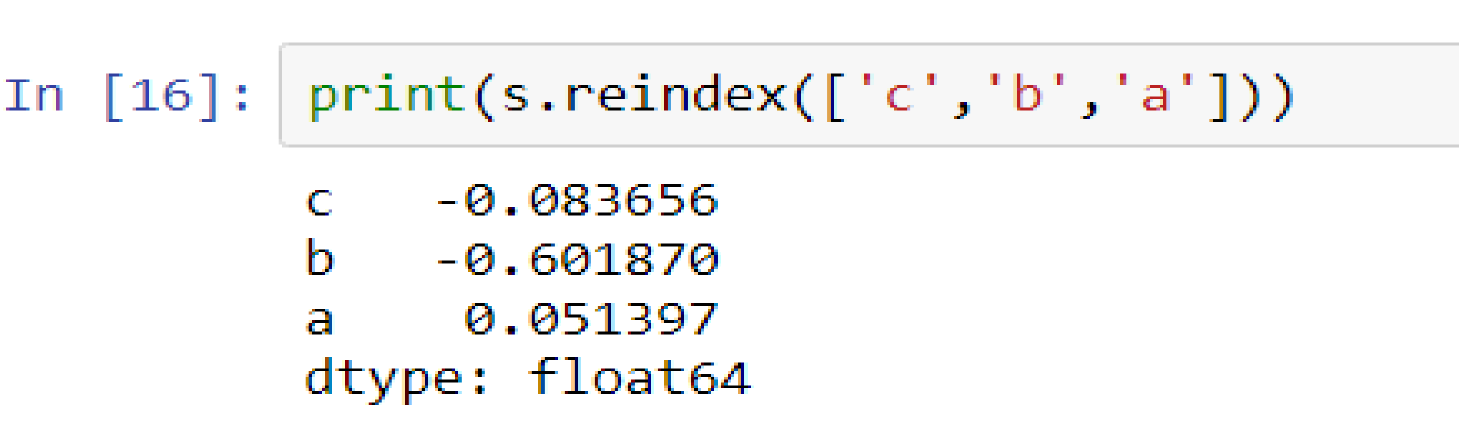
```
7
[RangeIndex(start=0, stop=7, step=1)]
float64
<pandas.core.indexing._AtIndexer object at 0x000001E040C67138>
[10. 14. 15. 25. 30. 45. 55.]
(7,)
0    10.0
1    14.0
dtype: float64
float64:dense
```

Series

Reindexing labels: changing indexing as per our choice



```
In [18]: import pandas as pd
         s=pd.Series((10,5,7),index=['a','b','c'])
         print(s.index)

         Index(['a', 'b', 'c'], dtype='object')
```

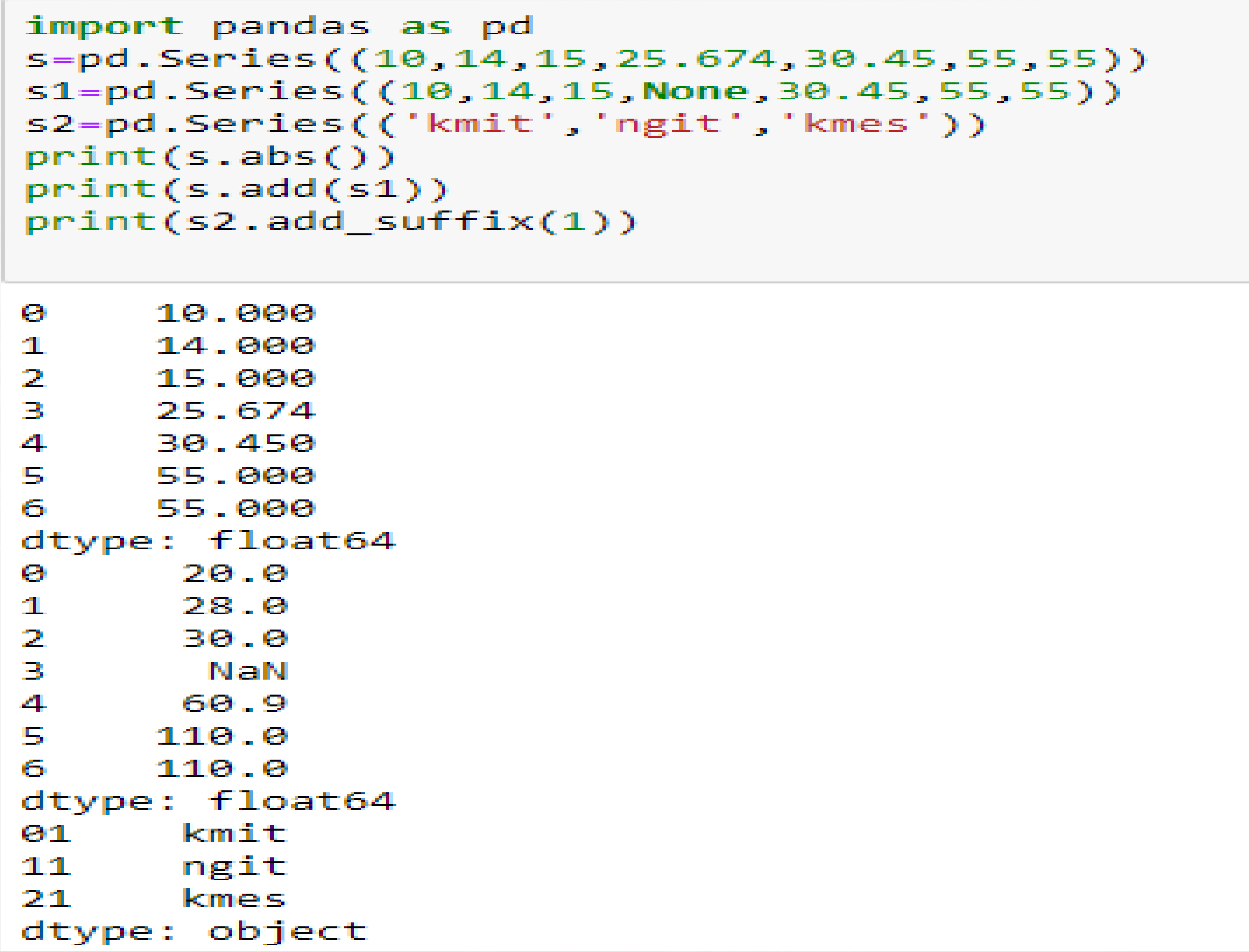


```
In [16]: print(s.reindex(['c','b','a']))

         c   -0.083656
         b   -0.601870
         a    0.051397
         dtype: float64
```

Series

Methods:

abs()
add()
add_suffix()



```
import pandas as pd
s=pd.Series((10,14,15,25.674,30.45,55,55))
s1=pd.Series((10,14,15,None,30.45,55,55))
s2=pd.Series(('kmit','ngit','kmes'))
print(s.abs())
print(s.add(s1))
print(s2.add_suffix(1))
```

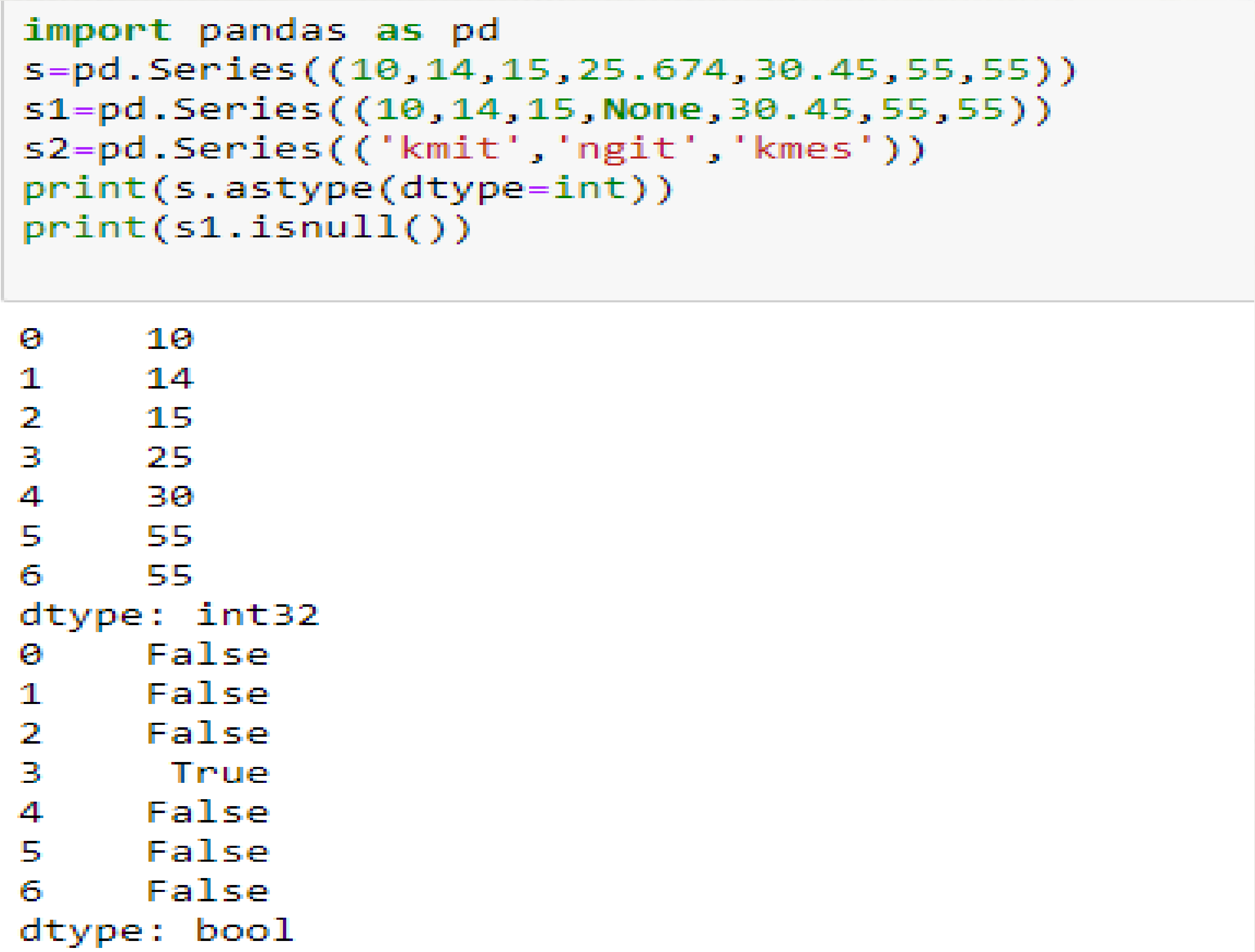
```
0     10.000
1     14.000
2     15.000
3     25.674
4     30.450
5     55.000
6     55.000
dtype: float64
0      20.0
1      28.0
2      30.0
3       NaN
4      60.9
5     110.0
6     110.0
dtype: float64
01     kmit
11     ngit
21     kmes
dtype: object
```

Series

Methods:

astype()

isnull()



```
import pandas as pd
s=pd.Series((10,14,15,25.674,30.45,55,55))
s1=pd.Series((10,14,15,None,30.45,55,55))
s2=pd.Series(('kmit','ngit','kmes'))
print(s.astype(dtype=int))
print(s1.isnull())
```

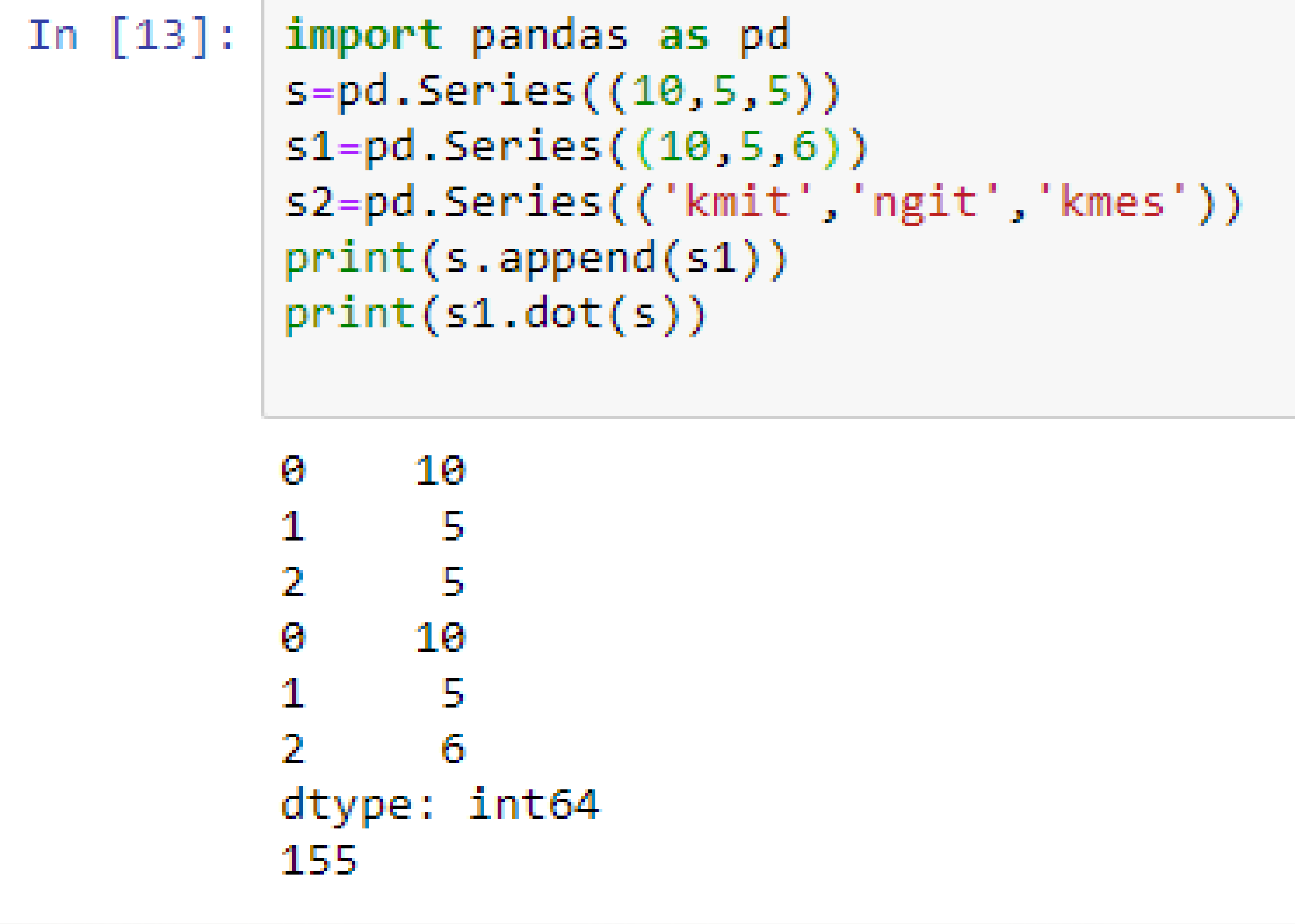
```
0     10
1     14
2     15
3     25
4     30
5     55
6     55
dtype: int32
0     False
1     False
2     False
3      True
4     False
5     False
6     False
dtype: bool
```

Series

Methods:

append()

dot()



```
In [13]: import pandas as pd
         s=pd.Series((10,5,5))
         s1=pd.Series((10,5,6))
         s2=pd.Series(('kmit','ngit','kmes'))
         print(s.append(s1))
         print(s1.dot(s))
```

```
0    10
1     5
2     5
0    10
1     5
2     6
dtype: int64
155
```

Series

Methods:

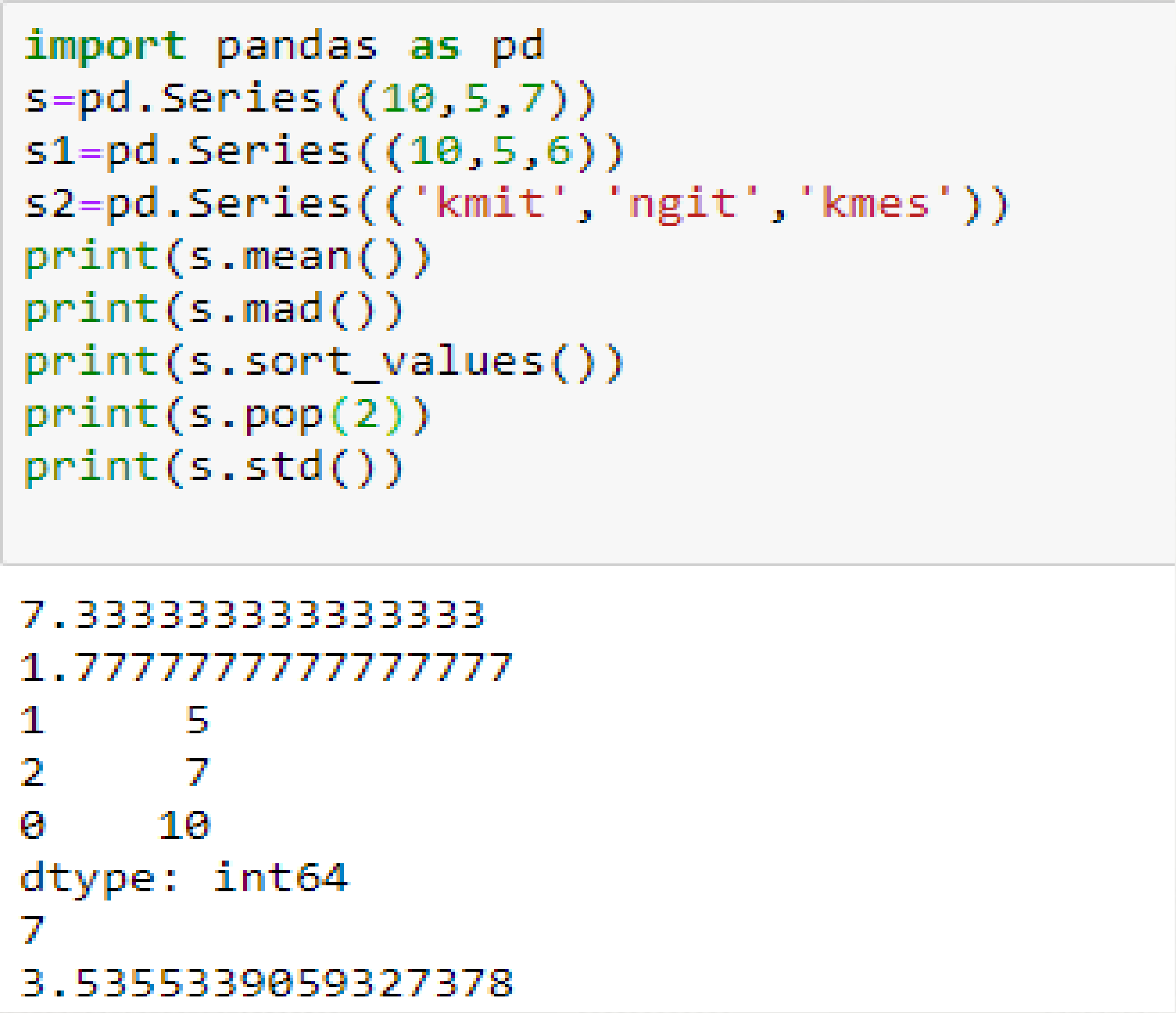
mean()

mad()

sort_values()

pop()

std()

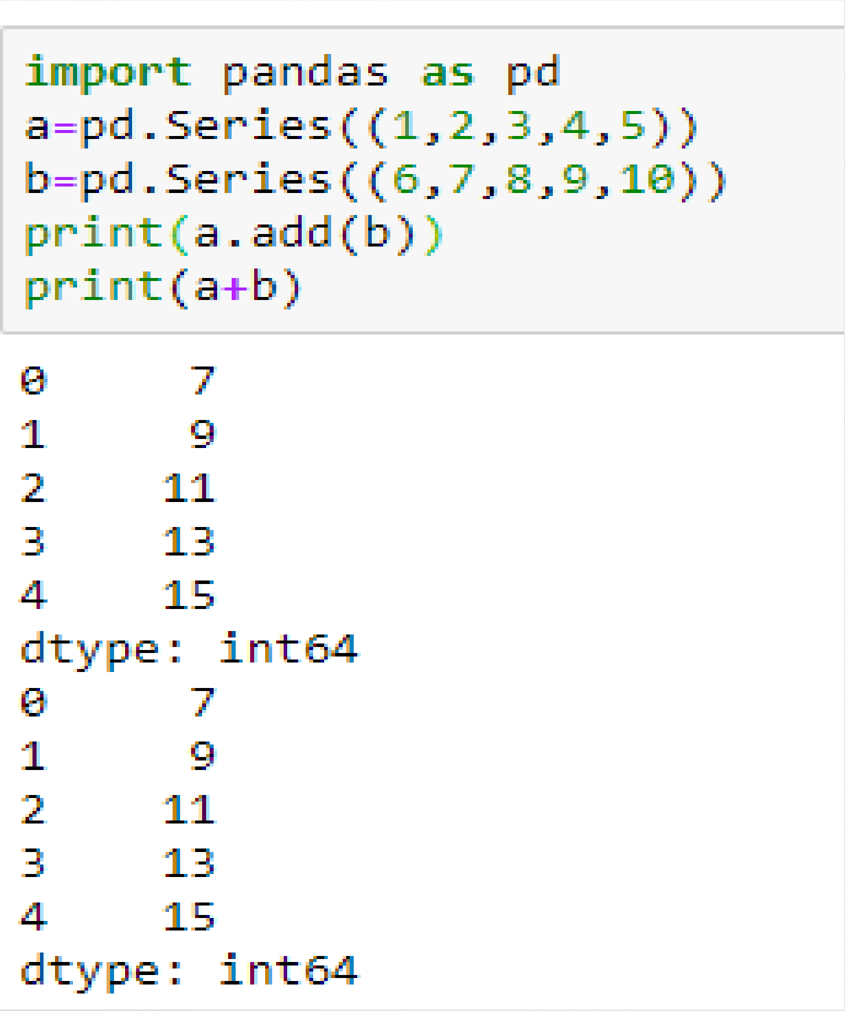


```
import pandas as pd
s=pd.Series((10,5,7))
s1=pd.Series((10,5,6))
s2=pd.Series(('kmit','ngit','kmes'))
print(s.mean())
print(s.mad())
print(s.sort_values())
print(s.pop(2))
print(s.std())
```

```
7.333333333333333
1.7777777777777777
1     5
2     7
0    10
dtype: int64
7
3.5355339059327378
```

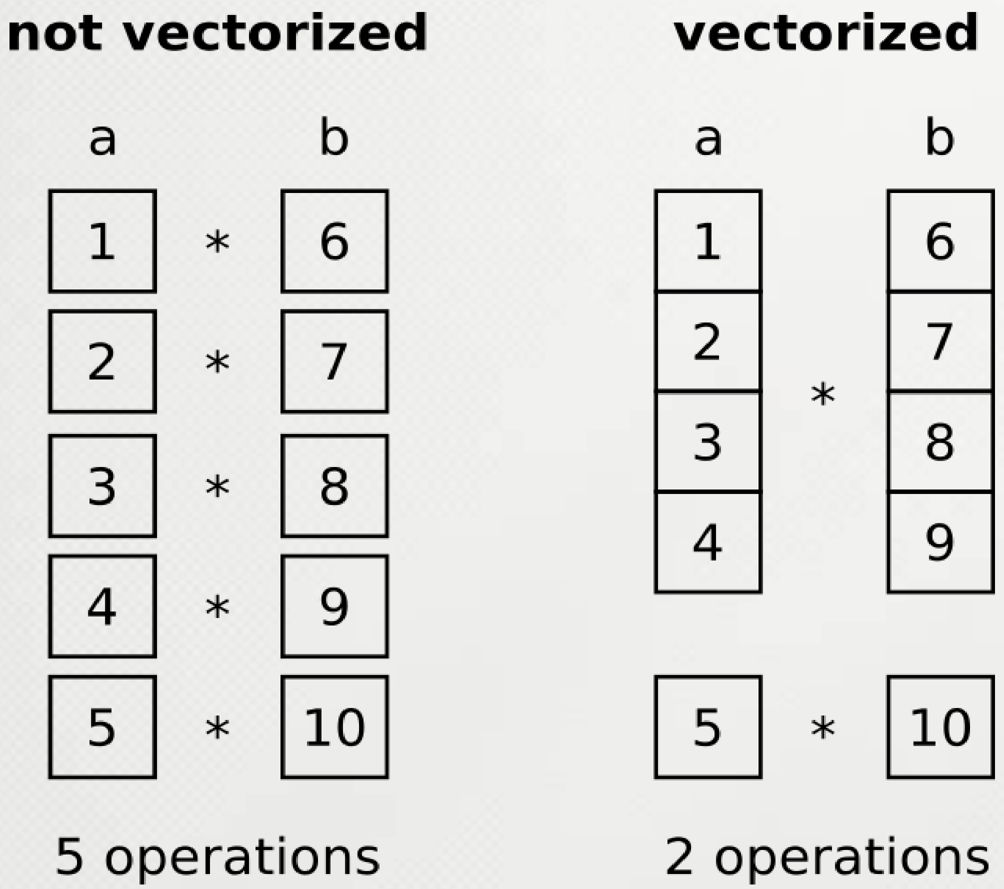
Vectorization

The process of rewriting a loop so that instead of processing a single element of an array N times.

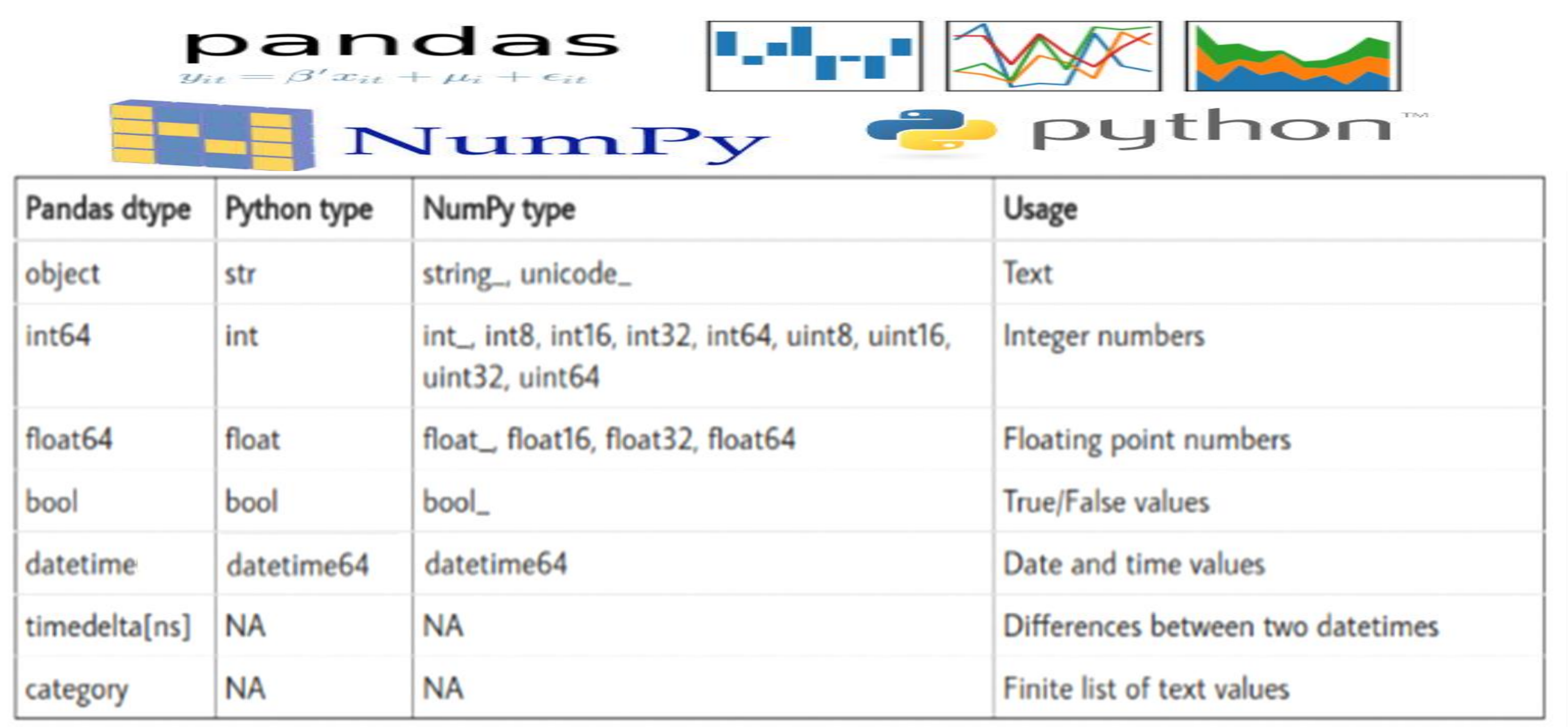


```
import pandas as pd
a=pd.Series((1,2,3,4,5))
b=pd.Series((6,7,8,9,10))
print(a.add(b))
print(a+b)
```

```
0     7
1     9
2    11
3    13
4    15
dtype: int64
0     7
1     9
2    11
3    13
4    15
dtype: int64
```



Pandas Data Types


pandas
$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$
NumPy
python™

Pandas dtype	Python type	NumPy type	Usage
object	str	string_, unicode_	Text
int64	int	int_, int8, int16, int32, int64, uint8, uint16, uint32, uint64	Integer numbers
float64	float	float_, float16, float32, float64	Floating point numbers
bool	bool	bool_	True/False values
datetime	datetime64	datetime64	Date and time values
timedelta[ns]	NA	NA	Differences between two datetimes
category	NA	NA	Finite list of text values

DataFrame

Two-dimensional tabular data structure

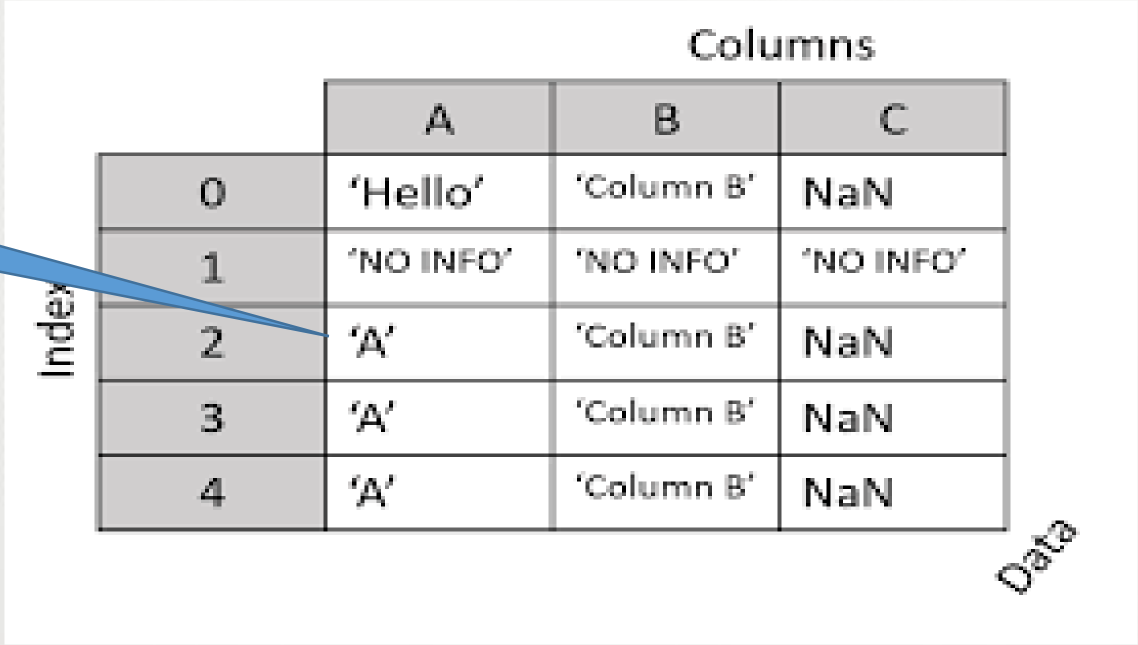
Data manipulation with integrated indexing

Support heterogeneous columns and Homogeneous columns

A pandas DataFrame – pandas. DataFrame(data, index, columns, dtype)

Data : ndarray\series\map\lists\dict\ DataFrame

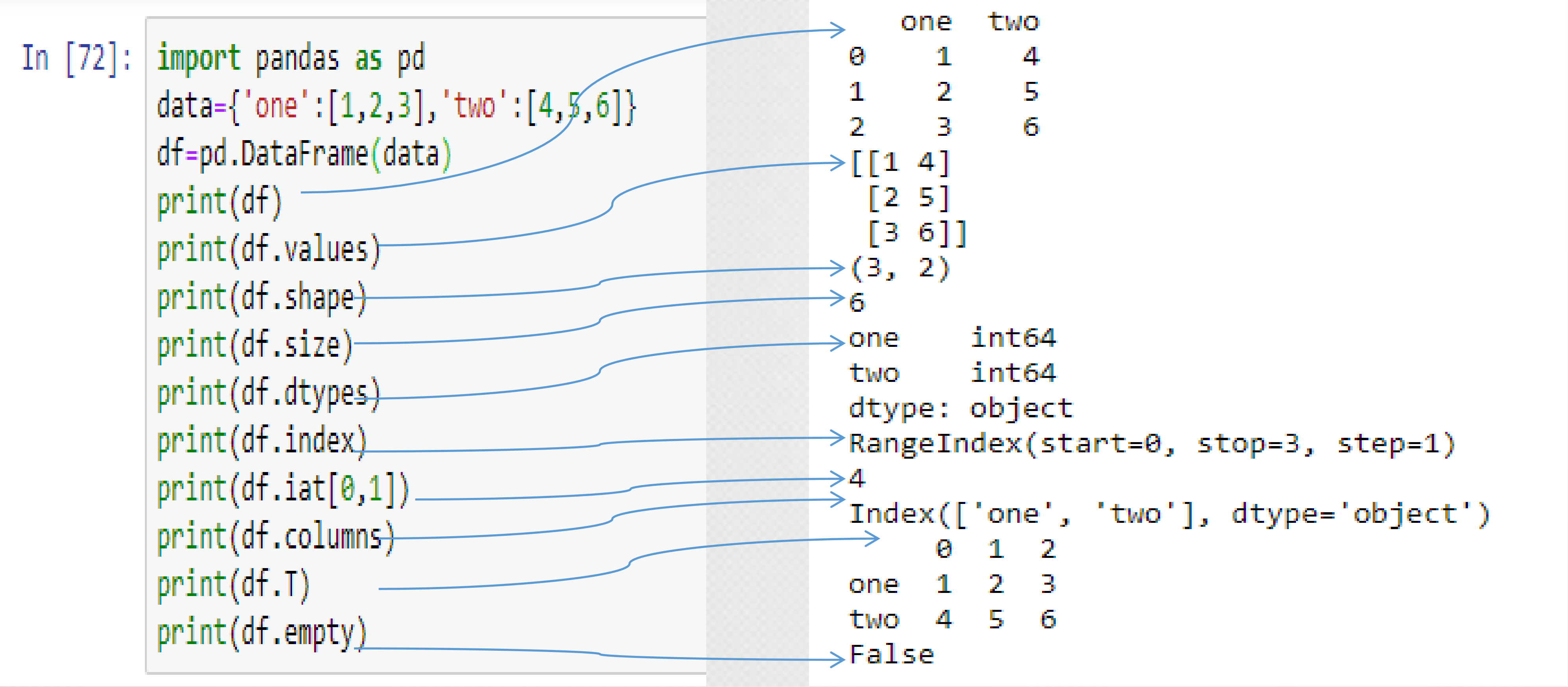
Dtype:-Data type of each column
Size:- Mutable


Columns
Index
Data

	A	B	C
0	'Hello'	'Column B'	NaN
1	'NO INFO'	'NO INFO'	'NO INFO'
2	'A'	'Column B'	NaN
3	'A'	'Column B'	NaN
4	'A'	'Column B'	NaN

DataFrame

Attributes:



```
In [72]: import pandas as pd
         data={'one':[1,2,3],'two':[4,5,6]}
         df=pd.DataFrame(data)
         print(df)
         print(df.values)
         print(df.shape)
         print(df.size)
         print(df.dtypes)
         print(df.index)
         print(df.iat[0,1])
         print(df.columns)
         print(df.T)
         print(df.empty)
```

```
   one  two
0    1    4
1    2    5
2    3    6
[[1 4]
 [2 5]
 [3 6]]
(3, 2)
6
one    int64
two    int64
dtype: object
RangeIndex(start=0, stop=3, step=1)
4
Index(['one', 'two'], dtype='object')
     0  1  2
one  1  2  3
two  4  5  6
False
```

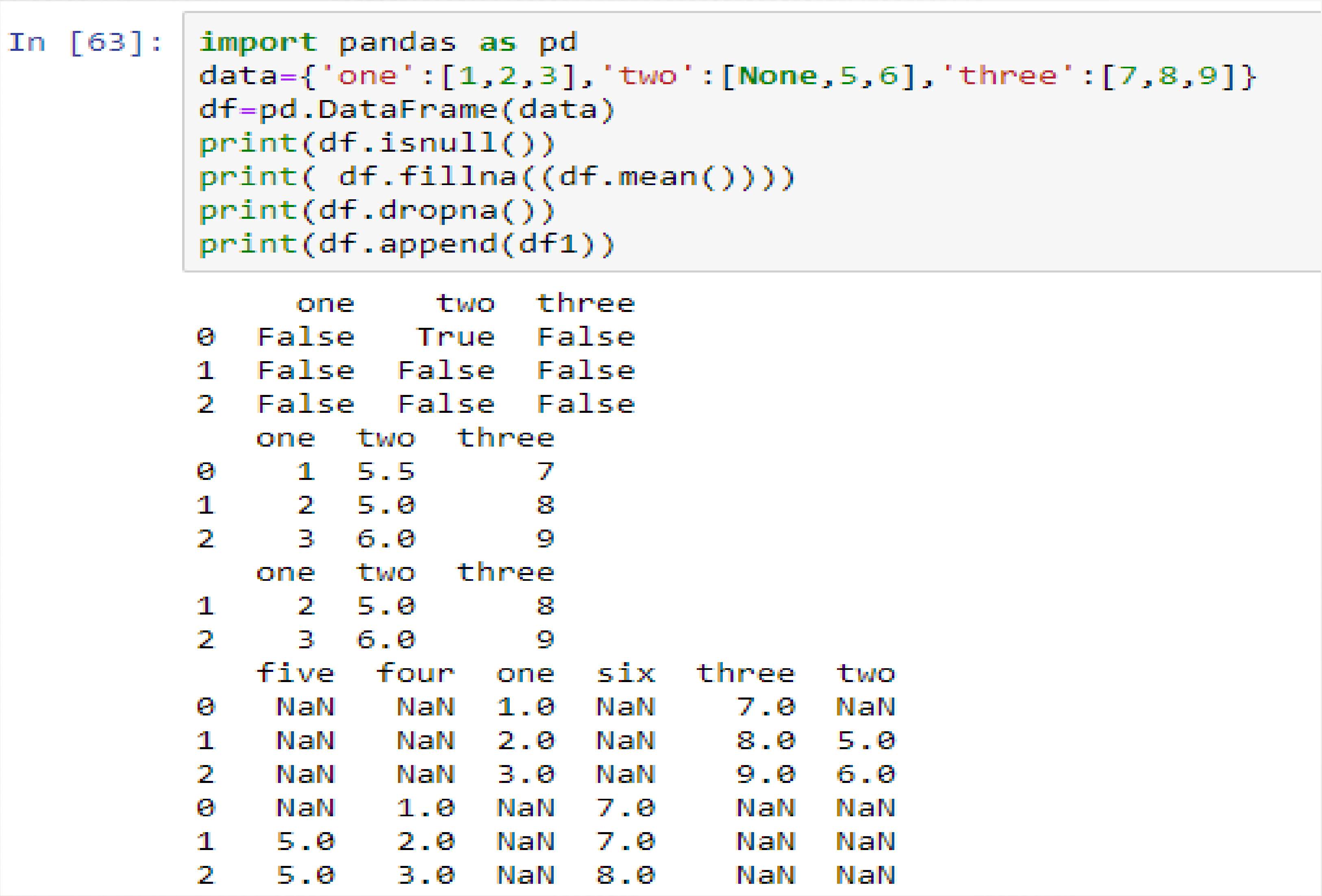
DataFrame

Methods:

isnull():   Detect missing values.

fillna([):Fill NA/NaN values using the specified method

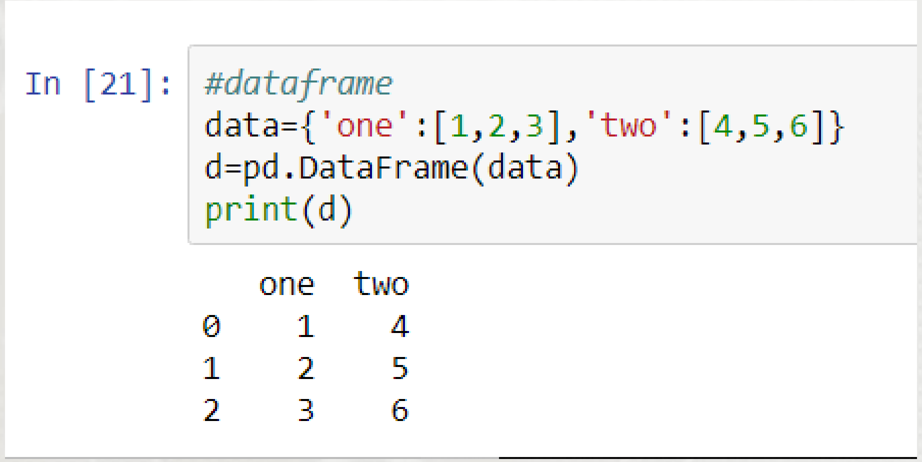
dropna():Remove missing values.



```
In [63]: import pandas as pd
         data={'one':[1,2,3],'two':[None,5,6],'three':[7,8,9]}
         df=pd.DataFrame(data)
         print(df.isnull())
         print( df.fillna((df.mean())))
         print(df.dropna())
         print(df.append(df1))
```

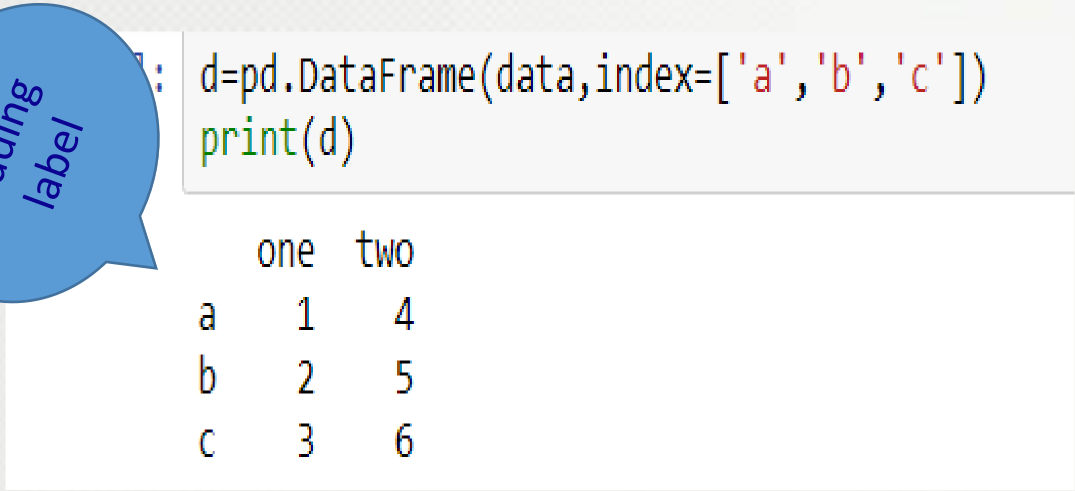
```
     one    two  three
0  False   True  False
1  False  False  False
2  False  False  False
   one  two  three
0    1  5.5      7
1    2  5.0      8
2    3  6.0      9
   one  two  three
1    2  5.0      8
2    3  6.0      9
   five  four  one  six  three  two
0   NaN   NaN  1.0  NaN    7.0  NaN
1   NaN   NaN  2.0  NaN    8.0  5.0
2   NaN   NaN  3.0  NaN    9.0  6.0
0   NaN   1.0  NaN  7.0    NaN  NaN
1   5.0   2.0  NaN  7.0    NaN  NaN
2   5.0   3.0  NaN  8.0    NaN  NaN
```

Creating DataFrame



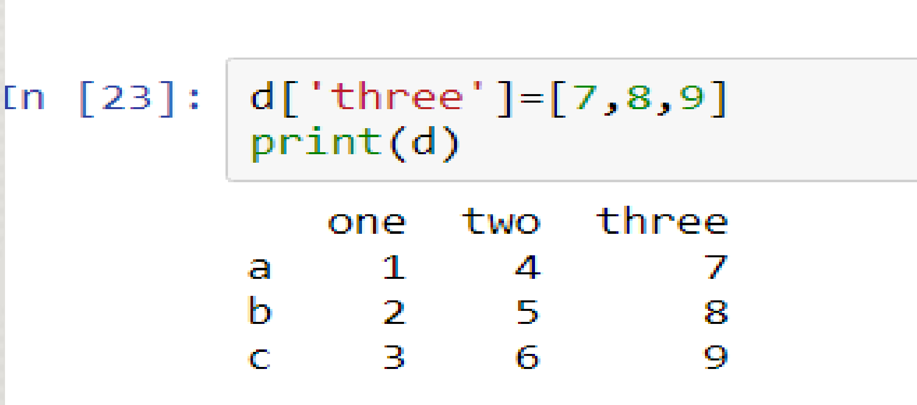
```
In [21]: #dataframe
         data={'one':[1,2,3],'two':[4,5,6]}
         d=pd.DataFrame(data)
         print(d)

            one  two
         0    1    4
         1    2    5
         2    3    6
```

Adding label



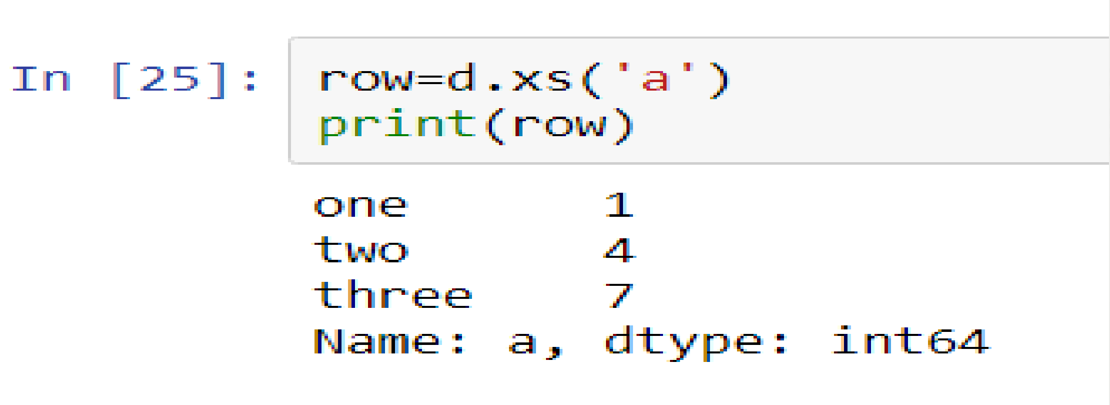
```
d=pd.DataFrame(data,index=['a','b','c'])
print(d)

   one  two
a    1    4
b    2    5
c    3    6
```

add column to DataFrame



```
In [23]: d['three']=[7,8,9]
         print(d)

            one  two  three
         a    1    4      7
         b    2    5      8
         c    3    6      9
```

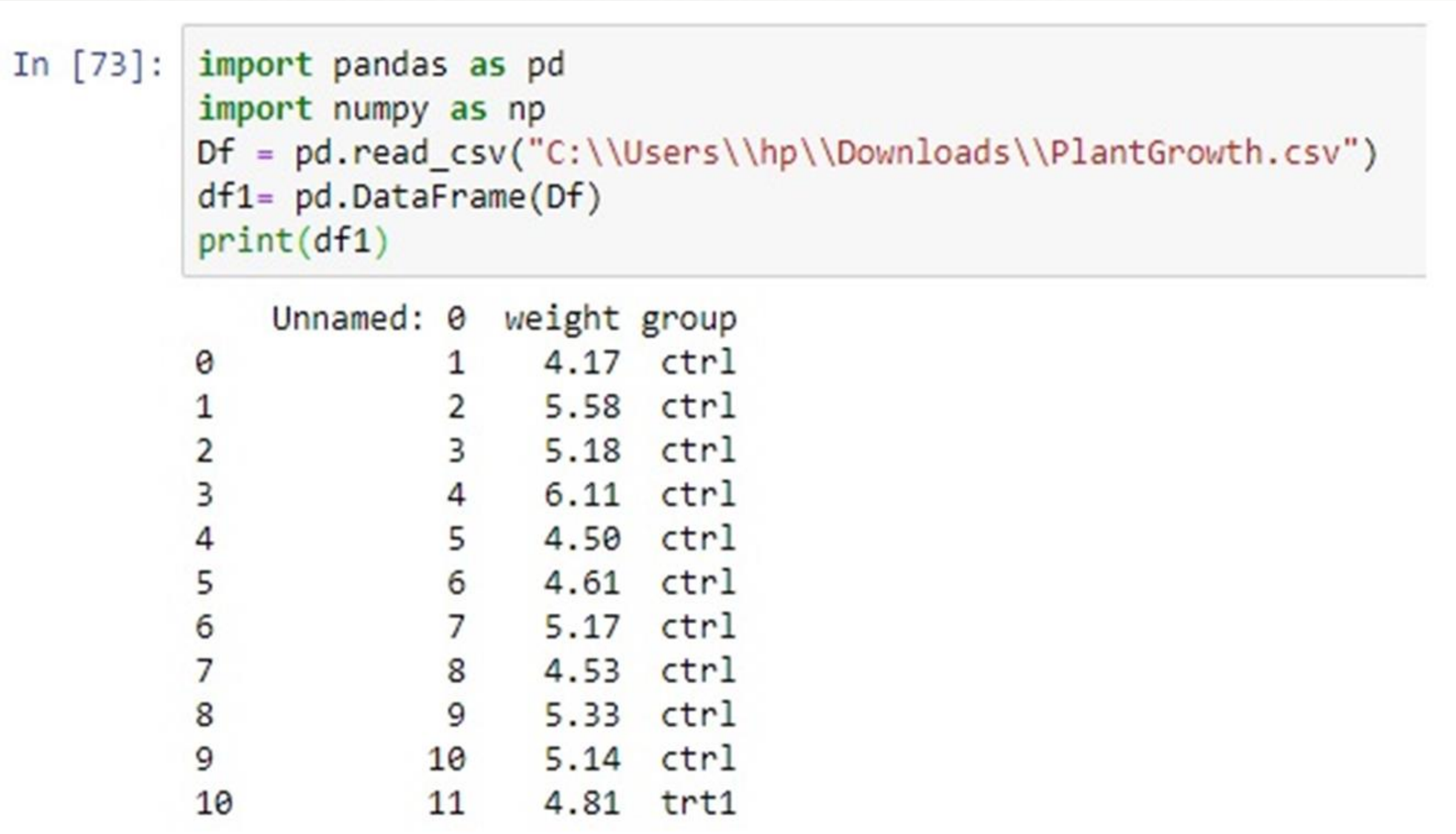
Select row by label



```
In [25]: row=d.xs('a')
         print(row)

         one      1
         two      4
         three    7
         Name: a, dtype: int64
```

Creating DataFrame

The Pandas I/O API is a set of top level reader functions accessed like pd.read_csv() that generally return a Pandas object.

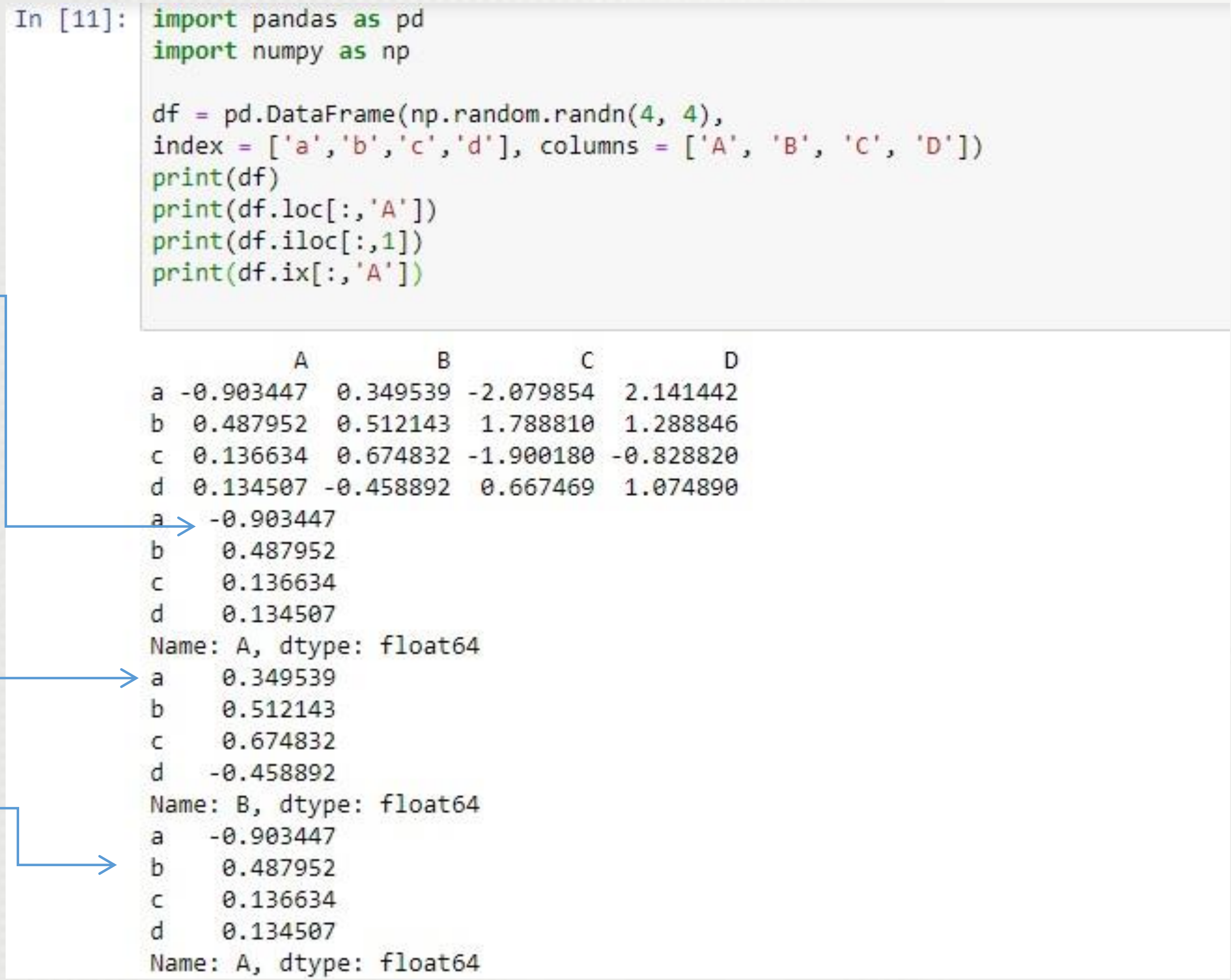


```
In [73]: import pandas as pd
         import numpy as np
         Df = pd.read_csv("C:\\Users\\hp\\Downloads\\PlantGrowth.csv")
         df1= pd.DataFrame(Df)
         print(df1)
```

```
    Unnamed: 0  weight group
0            1    4.17  ctrl
1            2    5.58  ctrl
2            3    5.18  ctrl
3            4    6.11  ctrl
4            5    4.50  ctrl
5            6    4.61  ctrl
6            7    5.17  ctrl
7            8    4.53  ctrl
8            9    5.33  ctrl
9           10    5.14  ctrl
10          11    4.81  trt1
```

Indexing and selecting Data

Indexing	Description
.loc()	Label based
.iloc()	Integer based
.ix()	Both Label & Integer based



```
In [11]: import pandas as pd
         import numpy as np

         df = pd.DataFrame(np.random.randn(4, 4),
         index = ['a','b','c','d'], columns = ['A', 'B', 'C', 'D'])
         print(df)
         print(df.loc[:,'A'])
         print(df.iloc[:,1])
         print(df.ix[:,'A'])
```

```
          A         B         C         D
a -0.903447  0.349539 -2.079854  2.141442
b  0.487952  0.512143  1.788810  1.288846
c  0.136634  0.674832 -1.900180 -0.828820
d  0.134507 -0.458892  0.667469  1.074890
a   -0.903447
b    0.487952
c    0.136634
d    0.134507
Name: A, dtype: float64
a    0.349539
b    0.512143
c    0.674832
d   -0.458892
Name: B, dtype: float64
a   -0.903447
b    0.487952
c    0.136634
d    0.134507
Name: A, dtype: float64
```

Group By

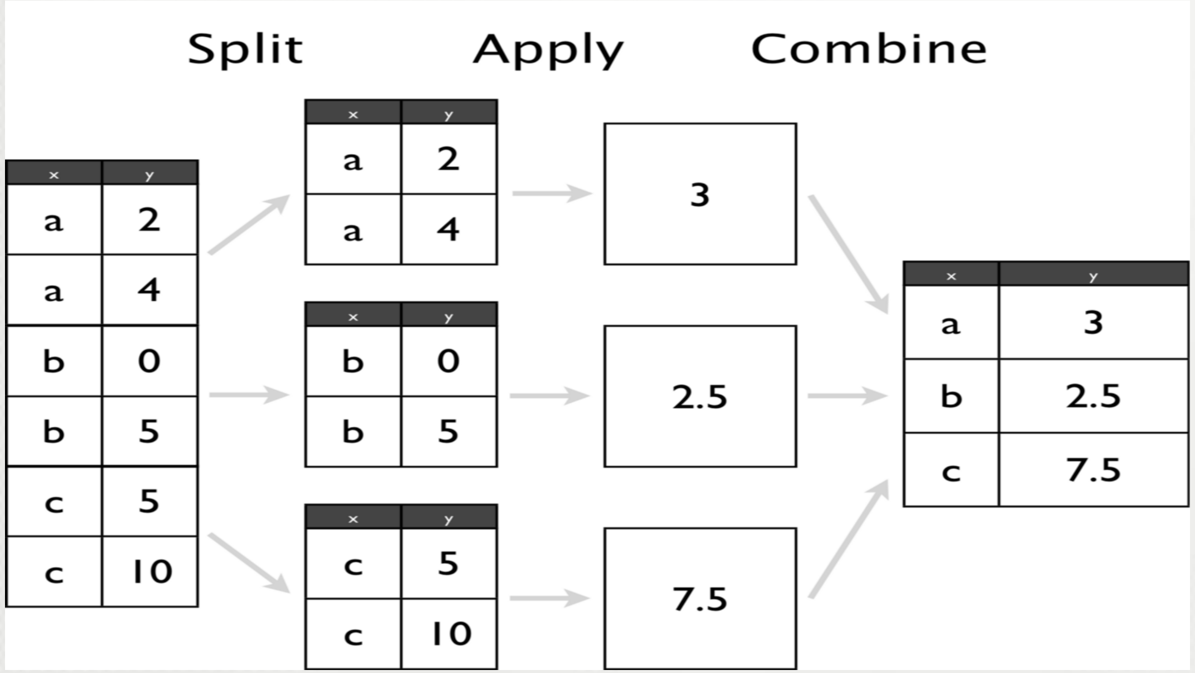
Any groupby operation involves one of the following operations on the original object.

They are –

Splitting the Object

Applying a function

Combining the results



Split Apply Combine

x	y
a	2
a	4
b	0
b	5
c	5
c	10

x	y
a	2
a	4

x	y
b	0
b	5

x	y
c	5
c	10

3

2.5

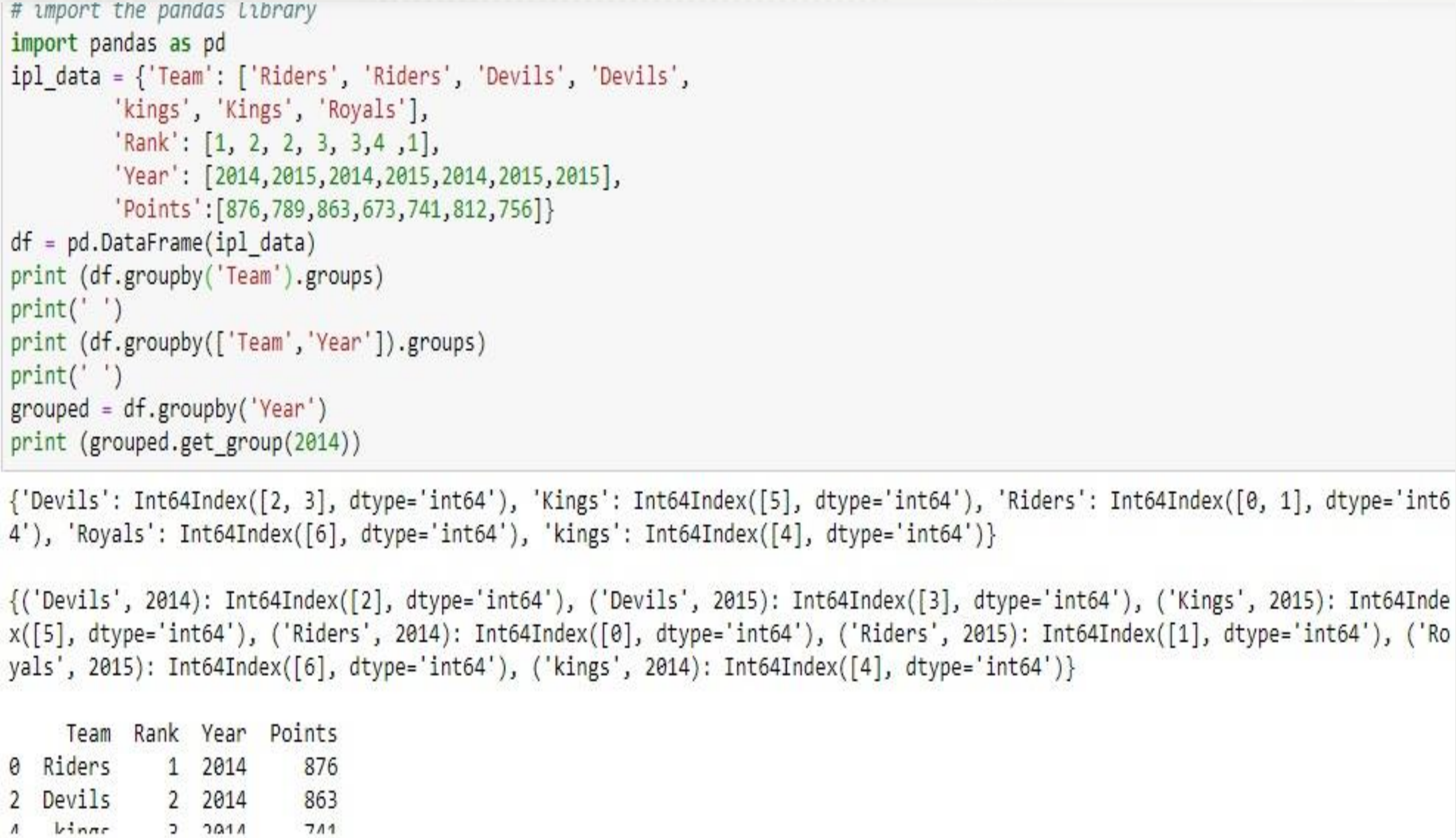
7.5

x	y
a	3
b	2.5
c	7.5

Split Data into Groups

There are multiple ways to split an object like –

groupby('key')
groupby(['key1','key2'])



```
# import the pandas library
import pandas as pd
ipl_data = {'Team': ['Riders', 'Riders', 'Devils', 'Devils',
         'kings', 'Kings', 'Royals'],
         'Rank': [1, 2, 2, 3, 3,4 ,1],
         'Year': [2014,2015,2014,2015,2014,2015,2015],
         'Points':[876,789,863,673,741,812,756]}
df = pd.DataFrame(ipl_data)
print (df.groupby('Team').groups)
print(' ')
print (df.groupby(['Team','Year']).groups)
print(' ')
grouped = df.groupby('Year')
print (grouped.get_group(2014))
```

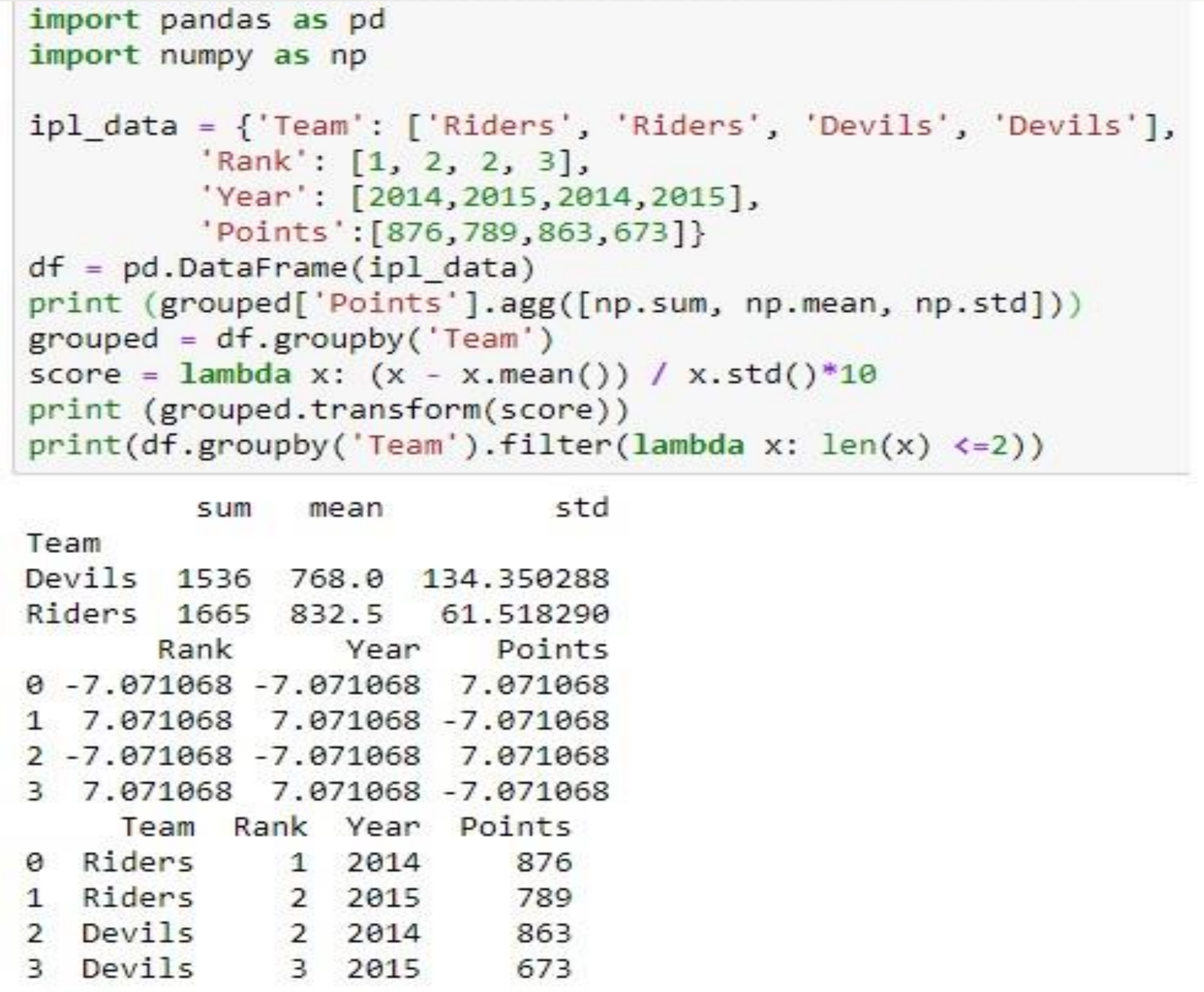
```
{'Devils': Int64Index([2, 3], dtype='int64'), 'Kings': Int64Index([5], dtype='int64'), 'Riders': Int64Index([0, 1], dtype='int6
4'), 'Royals': Int64Index([6], dtype='int64'), 'kings': Int64Index([4], dtype='int64')}

{('Devils', 2014): Int64Index([2], dtype='int64'), ('Devils', 2015): Int64Index([3], dtype='int64'), ('Kings', 2015): Int64Inde
x([5], dtype='int64'), ('Riders', 2014): Int64Index([0], dtype='int64'), ('Riders', 2015): Int64Index([1], dtype='int64'), ('Ro
yals', 2015): Int64Index([6], dtype='int64'), ('kings', 2014): Int64Index([4], dtype='int64')}

     Team  Rank  Year  Points
0  Riders     1  2014     876
2  Devils     2  2014     863
4   kings     3  2014     741
```

Applying Functions

Aggregation (Min, Max, Mode,Count,Var …etc)

Transformation

Filtration



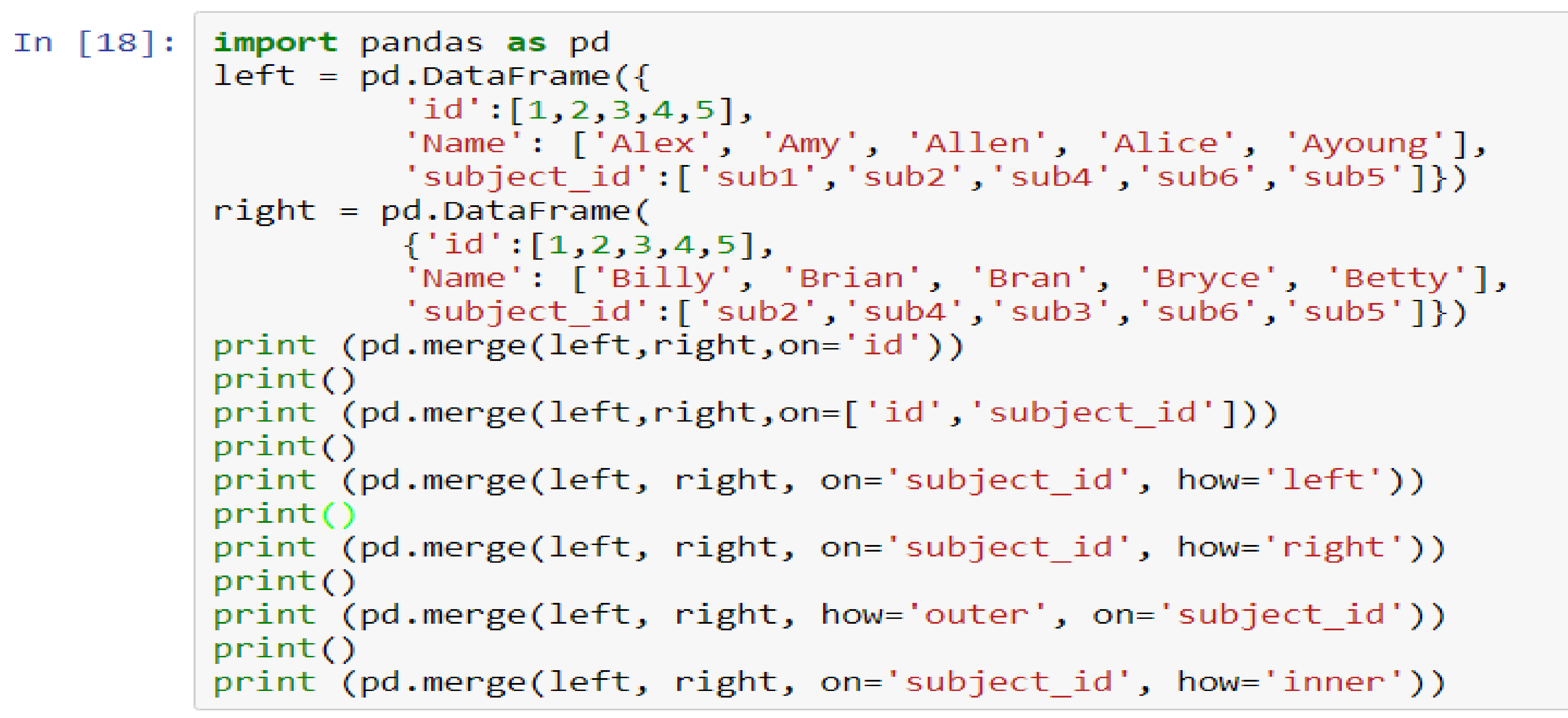
```
import pandas as pd
import numpy as np

ipl_data = {'Team': ['Riders', 'Riders', 'Devils', 'Devils'],
         'Rank': [1, 2, 2, 3],
         'Year': [2014,2015,2014,2015],
         'Points':[876,789,863,673]}
df = pd.DataFrame(ipl_data)
print (grouped['Points'].agg([np.sum, np.mean, np.std]))
grouped = df.groupby('Team')
score = lambda x: (x - x.mean()) / x.std()*10
print (grouped.transform(score))
print(df.groupby('Team').filter(lambda x: len(x) <=2))
```

```
         sum   mean         std
Team
Devils  1536  768.0  134.350288
Riders  1665  832.5   61.518290
       Rank      Year    Points
0 -7.071068 -7.071068  7.071068
1  7.071068  7.071068 -7.071068
2 -7.071068 -7.071068  7.071068
3  7.071068  7.071068 -7.071068
     Team  Rank  Year  Points
0  Riders     1  2014     876
1  Riders     2  2015     789
2  Devils     2  2014     863
3  Devils     3  2015     673
```

Merge & Joining

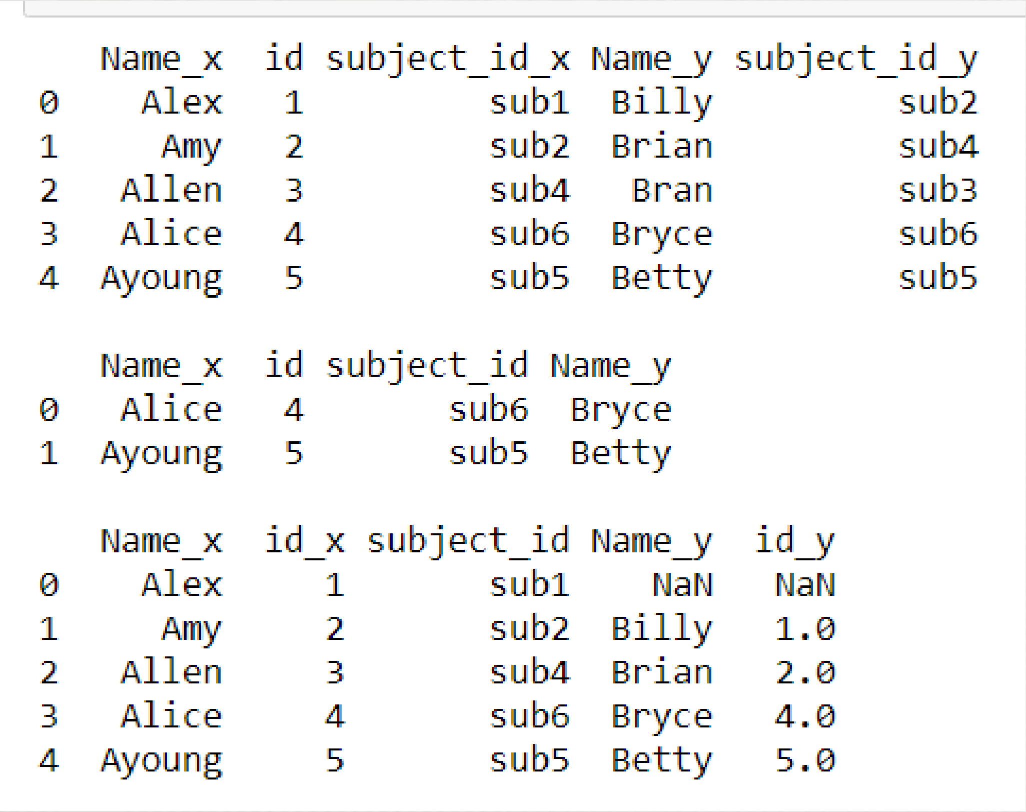
Merge Method	SQL Equivalent	Description
left	LEFT OUTER JOIN	Use keys from left object
right	RIGHT OUTER JOIN	Use keys from right object
outer	FULL OUTER JOIN	Use union of keys
inner	INNER JOIN	Use intersection of keys
sort	order	
how	set	– One of 'left', 'right', 'outer', 'inner'. Defaults to inner.

Merge & Joining

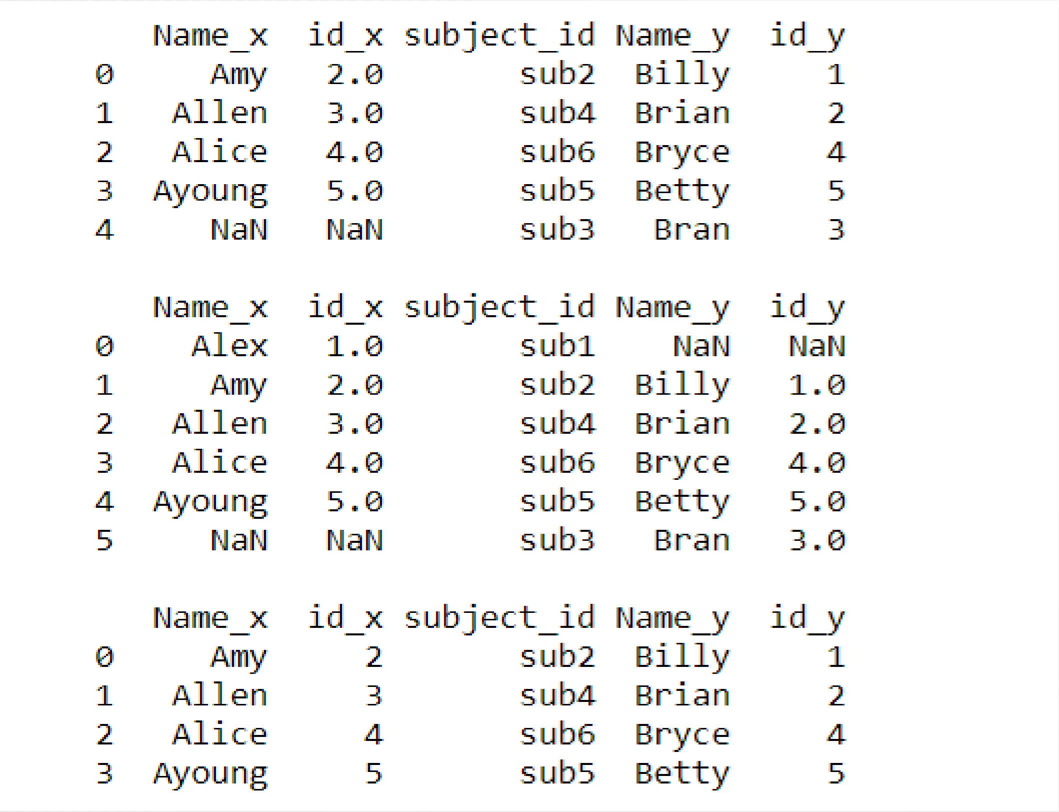


```
In [18]:
import pandas as pd
left = pd.DataFrame({
         'id':[1,2,3,4,5],
         'Name': ['Alex', 'Amy', 'Allen', 'Alice', 'Ayoung'],
         'subject_id':['sub1','sub2','sub4','sub6','sub5']})
right = pd.DataFrame(
         {'id':[1,2,3,4,5],
         'Name': ['Billy', 'Brian', 'Bran', 'Bryce', 'Betty'],
         'subject_id':['sub2','sub4','sub3','sub6','sub5']})
print (pd.merge(left,right,on='id'))
print()
print (pd.merge(left,right,on=['id','subject_id']))
print()
print (pd.merge(left, right, on='subject_id', how='left'))
print()
print (pd.merge(left, right, on='subject_id', how='right'))
print()
print (pd.merge(left, right, how='outer', on='subject_id'))
print()
print (pd.merge(left, right, on='subject_id', how='inner'))
```

Merge & Joining



```
   Name_x  id subject_id_x Name_y subject_id_y
0    Alex   1         sub1  Billy         sub2
1     Amy   2         sub2  Brian         sub4
2   Allen   3         sub4   Bran         sub3
3   Alice   4         sub6  Bryce         sub6
4  Ayoung   5         sub5  Betty         sub5

   Name_x  id subject_id Name_y
0   Alice   4       sub6  Bryce
1  Ayoung   5       sub5  Betty

   Name_x  id_x subject_id Name_y  id_y
0    Alex     1       sub1    NaN   NaN
1     Amy     2       sub2  Billy   1.0
2   Allen     3       sub4  Brian   2.0
3   Alice     4       sub6  Bryce   4.0
4  Ayoung     5       sub5  Betty   5.0
```



```
   Name_x  id_x subject_id Name_y  id_y
0     Amy   2.0       sub2  Billy     1
1   Allen   3.0       sub4  Brian     2
2   Alice   4.0       sub6  Bryce     4
3  Ayoung   5.0       sub5  Betty     5
4     NaN   NaN       sub3   Bran     3

   Name_x  id_x subject_id Name_y  id_y
0    Alex   1.0       sub1    NaN   NaN
1     Amy   2.0       sub2  Billy   1.0
2   Allen   3.0       sub4  Brian   2.0
3   Alice   4.0       sub6  Bryce   4.0
4  Ayoung   5.0       sub5  Betty   5.0
5     NaN   NaN       sub3   Bran   3.0

   Name_x  id_x subject_id Name_y  id_y
0     Amy     2       sub2  Billy     1
1   Allen     3       sub4  Brian     2
2   Alice     4       sub6  Bryce     4
3  Ayoung     5       sub5  Betty     5
```

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.

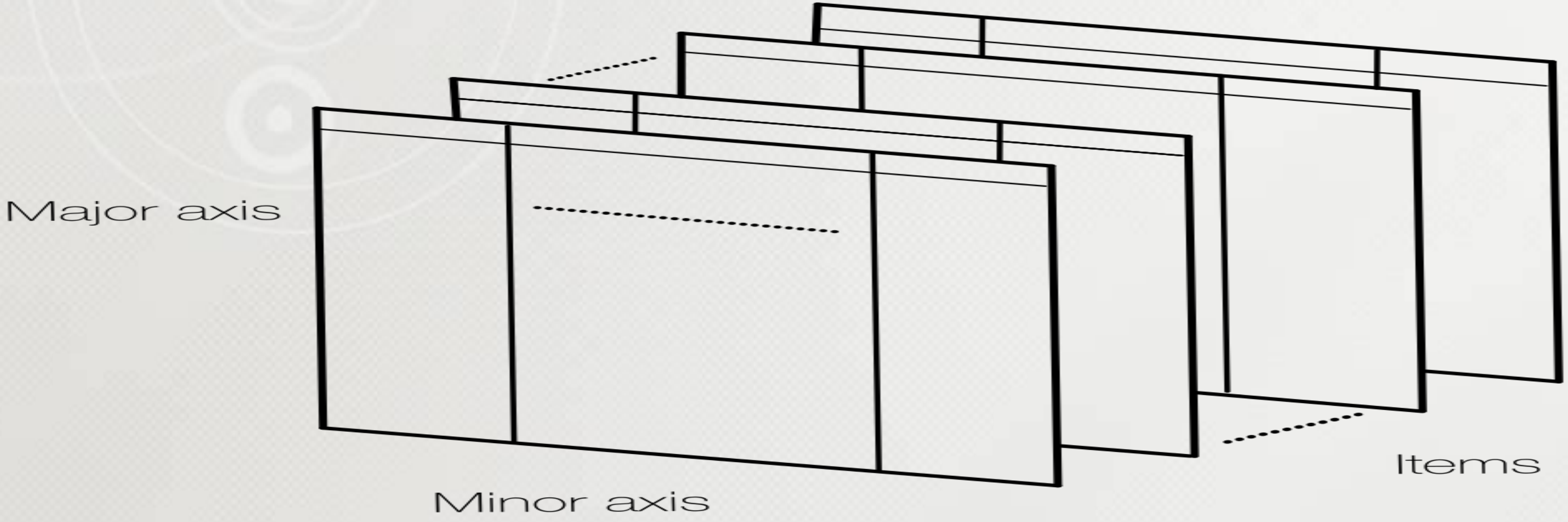
pandas. Panel(data, items, major axis, minor axis, dtype)

items – axis 0, each item corresponds to a DataFrame contained inside.

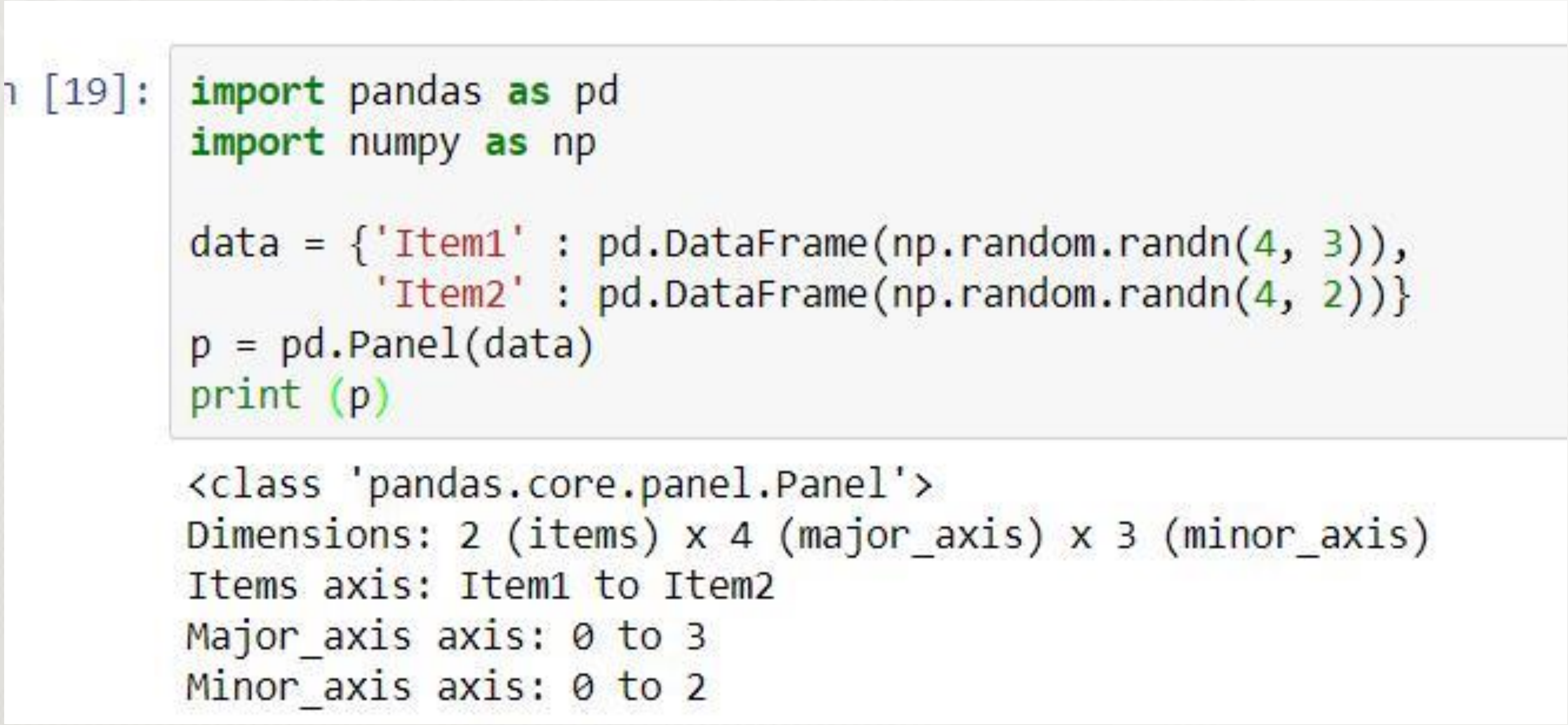
major_axis – axis 1, it is the index (rows) of each of the DataFrames.

Panel

minor_axis – axis 2, it is the columns of each of the DataFrames.



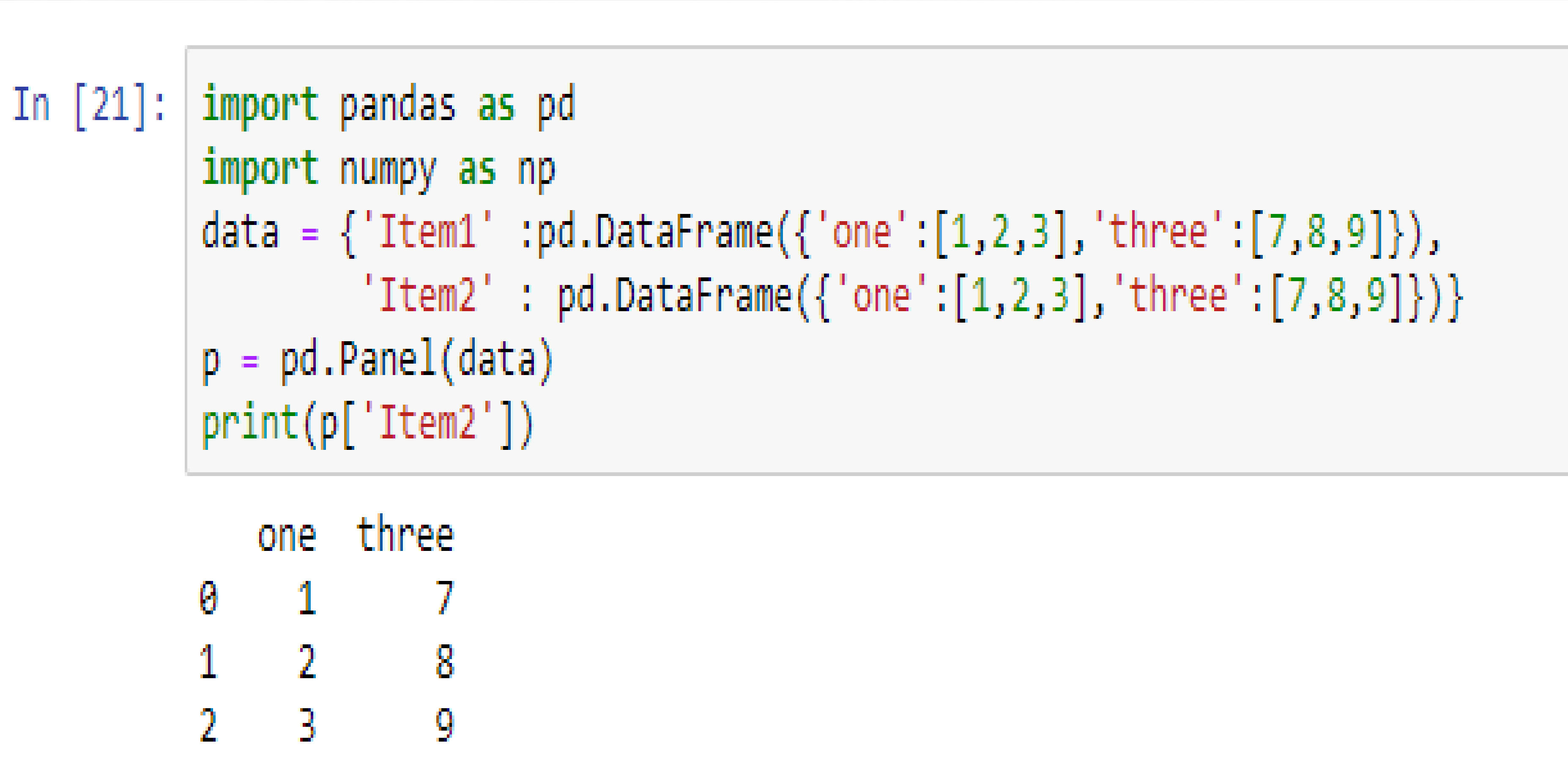
Panel



```
import pandas as pd
import numpy as np

data = {'Item1' : pd.DataFrame(np.random.randn(4, 3)),
        'Item2' : pd.DataFrame(np.random.randn(4, 2))}
p = pd.Panel(data)
print (p)
```

```
<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: 0 to 2
```

Panel



```
In [21]: import pandas as pd
         import numpy as np
         data = {'Item1' :pd.DataFrame({'one':[1,2,3],'three':[7,8,9]}),
                 'Item2' : pd.DataFrame({'one':[1,2,3],'three':[7,8,9]})}
         p = pd.Panel(data)
         print(p['Item2'])
```

```
   one  three
0    1      7
1    2      8
2    3      9
```

Panel

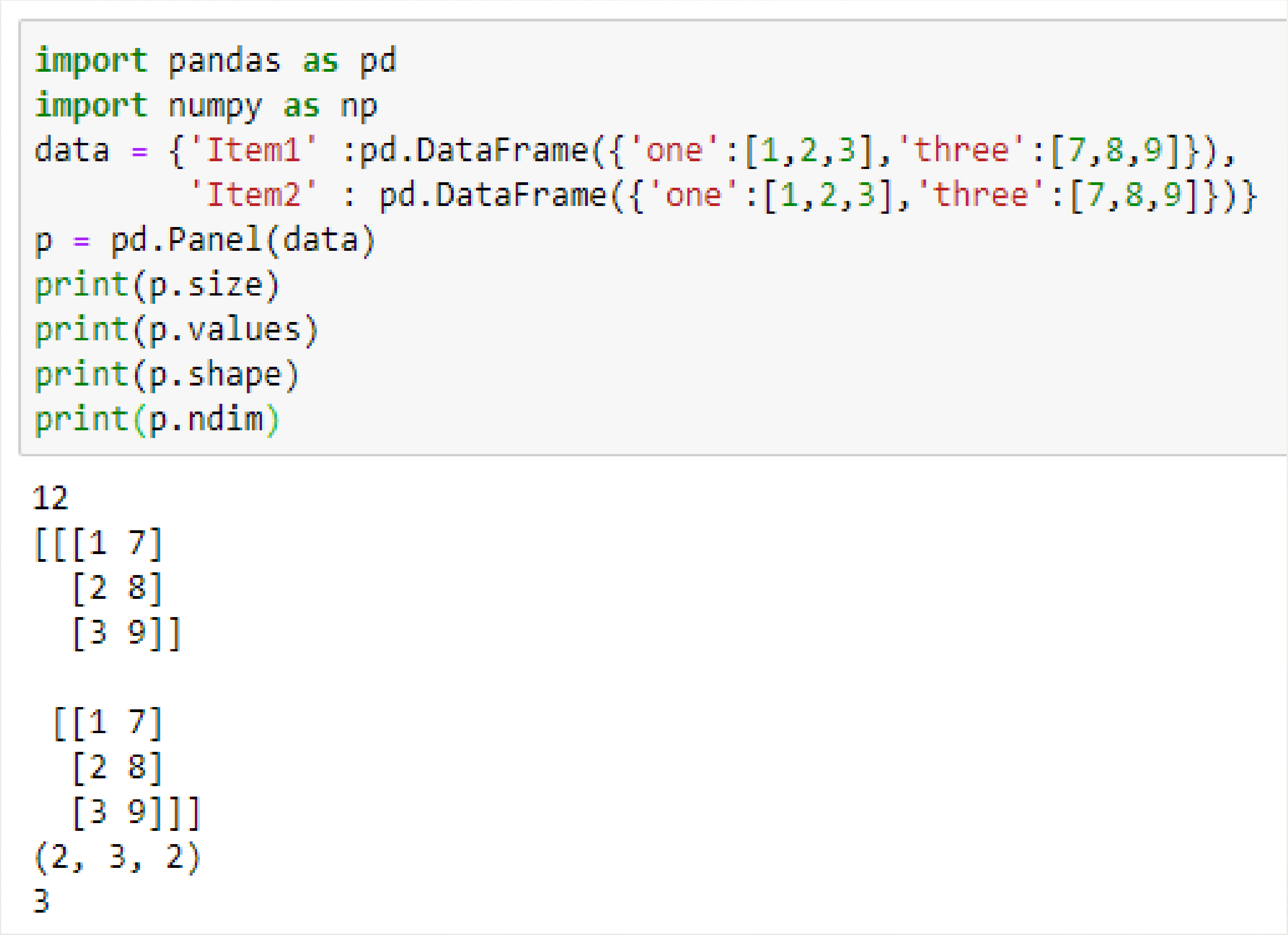
Attributes:

ndim: Return an int representing the number of axes / array dimensions.

shape: Return a tuple of axis dimensions

values: Return a Numpy representation of the DataFrame.

size: Return an int representing the number of elements in this object



```
import pandas as pd
import numpy as np
data = {'Item1' :pd.DataFrame({'one':[1,2,3],'three':[7,8,9]}),
        'Item2' : pd.DataFrame({'one':[1,2,3],'three':[7,8,9]})}
p = pd.Panel(data)
print(p.size)
print(p.values)
print(p.shape)
print(p.ndim)
```

```
12
[[[1 7]
  [2 8]
  [3 9]]

 [[1 7]
  [2 8]
  [3 9]]]
(2, 3, 2)
3
```

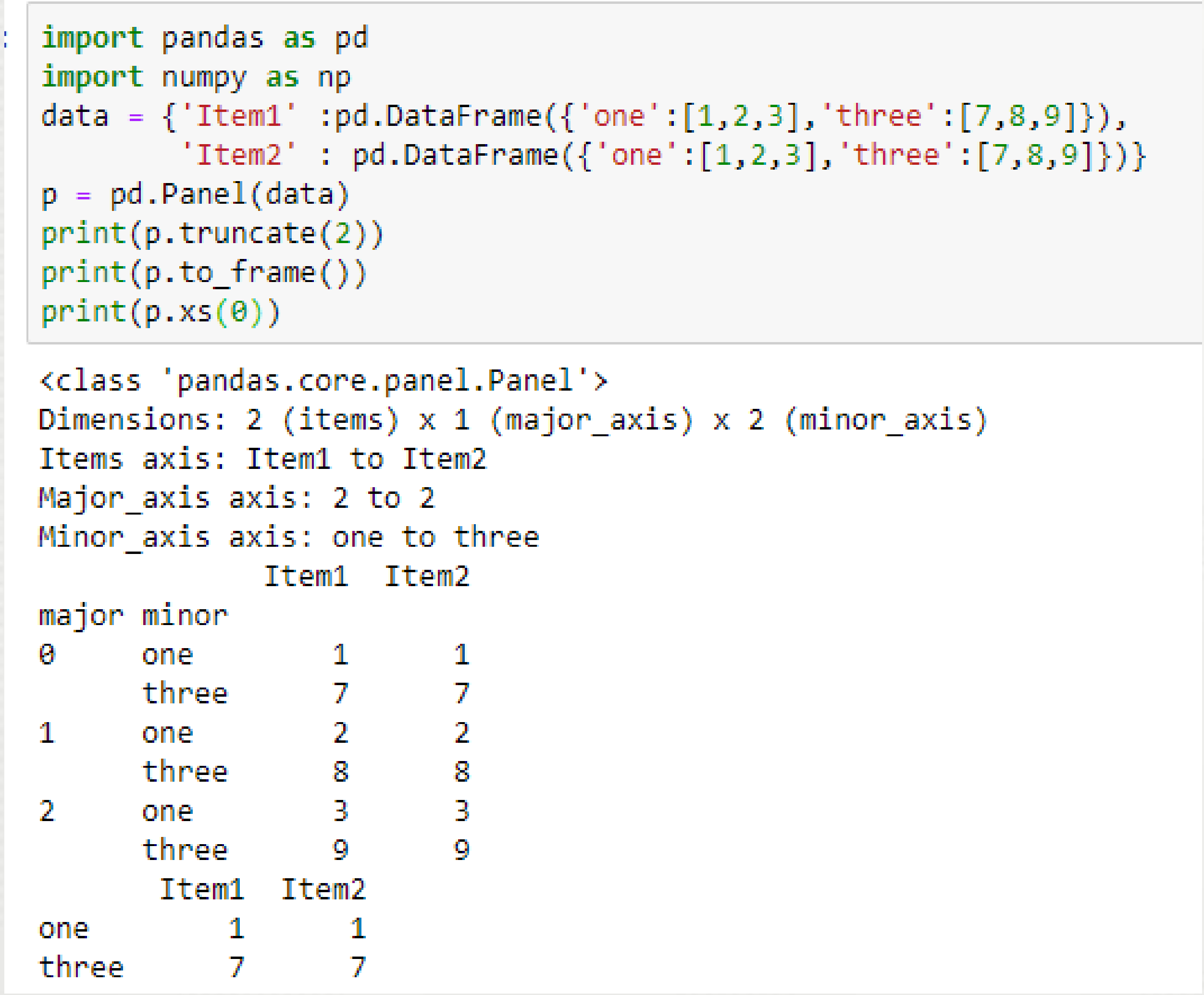
Panel

Methods:

truncate(): Truncate a Series or DataFrame before and after some index value.

xs(): Return slice of panel along selected axis

to_frame(): Round each value in Panel to a specified number of decimal places.



```
import pandas as pd
import numpy as np
data = {'Item1' :pd.DataFrame({'one':[1,2,3],'three':[7,8,9]}),
        'Item2' : pd.DataFrame({'one':[1,2,3],'three':[7,8,9]})}
p = pd.Panel(data)
print(p.truncate(2))
print(p.to_frame())
print(p.xs(0))
```

```
<class 'pandas.core.panel.Panel'>
Dimensions: 2 (items) x 1 (major_axis) x 2 (minor_axis)
Items axis: Item1 to Item2
Major_axis axis: 2 to 2
Minor_axis axis: one to three
             Item1  Item2
major minor
0     one        1      1
      three      7      7
1     one        2      2
      three      8      8
2     one        3      3
      three      9      9
       Item1  Item2
one        1      1
three      7      7
```

Conclusion

Discussed about ...

- Pandas– Types of data structures – creation-operations – methods

Next Session .....

Image and Audio Handling

THANK YOU